FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Acesso móvel a plataforma de e-Science para experiências de campo com sensores

Bruno Emanuel Cardoso Miranda Gonçalves

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. João Correia Lopes

23 de Fevereiro de 2018

Resumo

Em 2017, teve lugar uma campanha do projeto NEWA, no vale de Perdigão, concelho de Vila Velha de Rodão, com o propósito de recolher em micro-escala grande quantidade de dados em terrenos de orografia irregular, usando plataformas de medição de ponta. Nesta experiência avaliou-se uma resolução espacial de medidores em intervalos de décimos de segundos, na expectativa de se construirem modelos para o desenvolvimento de mapas de vento com resolução espacial de dezenas a centenas de metros para cobrir toda Europa.

Assim, com base num melhor conhecimento das condições de vento, poder-se-á quantificar o potencial energético eólico europeu e, através de conceção e operação de grandes turbinas, obter o aproveitamento eficiente dessa energia limpa. Em última análise, poder-se-á contribuir para a redução da grande dependência da energia fóssil.

A experiência de campo em Perdigão reveste-se da maior importância para os propósitos do projeto NEWA. São produzidas enormes quantidades de dados resultantes das experiências, simulações, sensores e satélites. A gestão de campanhas de campo, as análises dos dados e o arquivo das informações relevantes de cada campanha são conseguidos através do suporte da plataforma *e-Science* WindsP.

O principal objetivo deste trabalho é desenhar, implementar e testar uma aplicação móvel para gestão de eventos da campanha a decorrer em Perdigão.

Uma aplicação móvel num *smartphone* ou *tablet* possibilita a qualquer dos investigadores do projeto Perdigão fazer registos, ter fácil acesso às experiências desenvolvidas no perímetro experimental do Perdigão e acompanhar os resultados das investigações. As equipas poderão, também, tirar vantagens do fator mobilidade oferecido pelos dispositivos móveis, podendo aceder a qualquer hora e em qualquer lugar a todas as atividades que se desenrolam no terreno.

Um dos principais desafios no desenvolvimento de *software* em dispositivos móveis é o facto de se estar perante um cenário tecnológico altamente fragmentado e em rápida evolução. As aplicações para dispositivos móveis exigem uma quantidade razoável de personalização para serem executadas em diversas plataformas e responder ao fluxo contínuo de novos *hardwares*, versões de sistemas operativos e *browsers*. Optou-se por desenvolver uma aplicação que fosse capaz de funcionar em várias plataformas e adaptar-se às diferenças na tecnologia subjacente. Considerou-se a aplicação híbrida aquela que mais vantagens acumula para o caso em apreço.

A aplicação desenvolvida dá apoio aos investigadores que se encontram no terreno e permite tirar partido das funcionalidades do *smartphone* como o *Global Positioning System* (GPS) e câmara fotográfica. Com a aplicação e o auxílio do GPS, os investigadores podem pesquisar as estações que se encontram à sua volta e os dispositivos instalados em cada torre. Eles podem ainda, registar eventos, tais como: falhas de torres e equipamentos, condições físicas à volta das estações, e reportá-los imediatamente com a possibilidade de adicionar fotografias.

Palavras-chave: Experiência em Perdigão, *e-Science*, Aplicação móvel multi-plataforma.

Abstract

In 2017, a campaign of the NEWA project was carried out in the Perdigão valley, in the municipality of Vila Velha de Rodão, with the purpose of collecting a large amount of data in irregular terrain, using high-end measurement platforms. In this experiment, a spatial resolution of meters in tenth-second intervals was evaluated, with expectation of constructing models for the development of wind maps with spatial resolution of tens to hundreds of meters to cover all of Europe.

Thus, based on a better understanding of the wind conditions, it will be possible to quantify the European wind energy potential and, through the design and operation of large turbines, to obtain the efficient use of this clean energy. Ultimately, it will be possible to reduce the high dependence on fossil energy.

The Perdigão experiments are of the utmost importance for the purposes of the NEWA project. Huge amounts of data are produced resulting from experiments, simulations, sensors and satellites. Field campaign management, data analysis and the archiving of relevant information for each campaign are achieved through support of the WindsP e-Science platform.

The main objective of our work is to design, implement and test a mobile application for event management of the ongoing campaign in Perdigão.

A mobile application on a smartphone or tablet makes it possible for any of the Perdigão project researchers to record, to have easy access to the experiments developed in Perdigão's experimental perimeter and to follow the investigations. The teams can also take advantage of the mobility factor offered by mobile devices and can access anytime, anywhere for all the activities that take place in the field.

One of the main challenges in the development of mobile software is the highly fragmented and rapidly evolving technological scenario. Mobile applications require a fair amount of customization to run on multiple platforms and respond to the continuous flow of new hardware, operating system versions, and browsers. It was decided to develop an application that would be able to work across platforms and adapt to differences in the underlying technology. The hybrid application was considered the most advantageous for the present case.

The application supports the researchers in the field and take advantage of the features of the smartphone as the Global Positioning System (GPS) and camera. With the application and help of the GPS, the researchers will be able to search the stations that are around them and the respective devices installed in each tower. They may record events such as tower failures and equipment, physical conditions around the stations, and report them immediately with eventual photo integration.

Keywords: Experiment in Perdigão, e-Science, Cross-platform mobile application.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	*Application Programming Interface*
CSV	*Comma-Separated Values*
CLI	*Command Line Interface*
CRUD	*Create, Read, Update, Delete*
CSS	*Cascading Style Sheets*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
JPG	*Joint Photographic Experts Group*
JSON	*JavaScript Object*
KML	*Keyhole Markup Language*
LiDAR	*Light Detection And Ranging*
MVC	*Model View Controller*
MVVM	*Model View ViewModel*
MVW	*Model View Whatever*
NEWA	*New European Wind Atlas*
PDF	*Portable Document Format*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
SO	*Sistema Operativo*
SPA	*Single Page Application*
UI	*User Interface*
URL	*Uniform Resource Locator*
UX	*User eXperience design*

Capítulo 1

Introdução

Neste capítulo é apresentado o contexto do presente trabalho, seguido da motivação que levou à proposta do mesmo, bem como, dos objetivos que se pretendem atingir. Por fim, é apresentada a estrutura do documento.

1.1 Âmbito

O presente trabalho surge no âmbito de um projeto do consórcio *New European Wind Atlas* (NEWA), que visa produzir uma base de dados com as características do vento na Europa e uma nova geração de modelos de fluxo para explorar essa mesma base de dados [1].

1.2 Contexto e Motivação

Nos meados do século XX, os dados produzidos nas investigações científicas eram mantidos em cadernos científicos e/ou discos magnéticos, tornando difícil o seu acesso e reutilização por parte de outros indivíduos e instituições [2].

Com a evolução da tecnologia, nomeadamente o aumento das capacidades dos sistemas de computação, armazenamento e *softwares*, alterou-se a forma como a investigação científica é feita [3]. Passou-se a produzir enormes quantidades de dados através de experiências, simulações, sensores e satélites, tornando-se difícil a sua análise, sendo necessário plataformas que permitam a gestão dos dados e apoio aos investigadores. Esse facto resulta numa mudança de paradigma, de trabalhar sozinho com os seus próprios dados, para trabalho em equipa com dados de várias fontes e, finalmente, a trabalhar em equipas distribuídas e com grandes conjuntos de dados [4]. A plataforma *e-Science* (WindsP) de suporte ao planeamento, gestão de campanhas de campo, análise dos dados e arquivo das informações relevantes de cada campanha é utilizada pelos investigadores do projeto NEWA [5].

M. Riedel define *e-Science* como "uma nova área de investigação, com base na cooperação em áreas-chave da ciência e que utiliza uma infraestrutura de inovação, a fim de expandir os poderes da computação científica" [6].

De acordo com Shannon Bohle, "*e-Science* é a aplicação de tecnologias de computação para a realização de investigação científica moderna, incluindo a preparação, a experimentação, a recolha de dados, divulgação de resultados, armazenamento a longo prazo e acessibilidade a todos os materiais produzidos através do processo científico" [7].

O projeto NEWA tem os seguintes objetivos [1]:

- Contribuir para uma redução significativa do custo da energia eólica através da atenuação dos riscos associados à conceção e operação de grandes turbinas eólicas com base num melhor conhecimento das condições de vento;
- Quantificar melhor o potencial energético eólico europeu e fornecer dados e modelos que possam melhorar as ferramentas e operações de ordenamento do território, assegurando um desdobramento eficaz e eficiente da energia eólica.

Para atingir esses objetivos torna-se necessário levar a cabo a preparação e execução de experiências (campanhas de campo). Desse modo são utilizados, entre outros, sensores laser (LiDAR) para a análise e registo das condições de vento em instalações de torres eólicas de produção de energia elétrica.

Em 2017 teve lugar uma campanha do projeto NEWA no vale de Perdigão no concelho de Vila Velha de Rodão. A campanha de Perdigão envolve uma quantidade significativa de equipamentos como torres meteorológicas (50), sensores e anemómetros sónicos (344), *Scanning* Lidars (21), *profiling* Lidars (7), radiossondas (112). Estão envolvidos 8 países com dezenas de participantes incluindo universidades, empresas e laboratórios de pesquisa e mais de 70 técnicos de campo [8]. Este projeto vem ajudar a encontrar uma solução para aquele que continua sendo o grande desafio do mundo, a questão energética. O aproveitamento da energia ainda não atingiu um nível satisfatório, visto que a maioria da energia utilizada no planeta é, ainda, de origem não renovável, seja de fonte mineral ou atómica. Segundo Jakob Mann, a Europa obtém 11% da energia total do vento. Mas apenas uma mudança de 10% na velocidade do vento pode alterar a quantidade de energia produzida em até 30% [9].

A União Europeia pretende aumentar a participação de energia eólica na produção total da energia e, por isso, a campanha de Perdigão reveste-se de grande importância. A ambição é passar a participação dos atuais 16,5% para 25% até 2030. Os Estados Unidos, no horizonte 2030, também ambicionam aumentar sua participação em 20% acima dos atuais 6% [10].

No processo de coleta em micro-escala de grande quantidade de dados em terrenos de orografia variada, usando plataformas de medição de ponta e ao mesmo tempo que se avalia uma resolução espacial de medidores e intervalos de décimos de segundos, a campanha de Perdigão representa um progresso enorme. A expectativa dos investigadores é poderem construir modelos para o desenvolvimento de mapas de vento com resolução espacial de dezenas a centenas de metros para cobrir toda a Europa [10].

A nossa motivação é, aproveitando as potencialidades dos dispositivos móveis, portabilidade e disponibilidade, criar uma aplicação móvel que permita aos investigadores no terreno obter dados

da campanha em qualquer hora e sítio, podendo também tirar proveito dos recursos nativos do dispositivo móvel tais como sensores, câmara fotográfica, lista de contactos, acesso *offline*, receber notificações, etc.

1.3 Objetivos

O principal objetivo deste trabalho é desenhar, implementar e testar uma aplicação móvel de gestão de eventos para a campanha a decorrer em Perdigão. A aplicação dará apoio aos investigadores que se encontram no terreno e permitirá tirar partido das funcionalidades do *smartphone* como o *Global Positioning System* (GPS) e a câmara fotográfica. Com a aplicação e o auxílio do GPS, o investigador poderá pesquisar as estações que se encontram à sua volta e os respetivos dispositivos instalados em cada torre. Poderá registar eventos tais como falhas de torres e equipamentos, condições físicas à volta das estações e reportá-los imediatamente com eventual integração de fotografias.

1.4 Estrutura do Documento

Para além da Introdução, este documento contém mais seis capítulos, com a seguinte estrutura:

- O Capítulo 2, Revisão Bibliográfica, apresenta um estudo das abordagens do desenvolvimento de aplicações, *frameworks* de desenvolvimento e a plataforma *e-Science* WindsP.
- O Capítulo 3, Análise de Requisitos, apresenta a descrição do problema e os requisitos da aplicação.
- O Capítulo 4, Arquitetura, apresenta a solução do problema e as escolhas tecnológicas.
- O Capítulo 5, Implementação, apresenta os aspetos da implementação.
- O Capítulo 6, Resultados, apresenta os resultados obtidos.
- O Capítulo 7, Conclusão, apresenta as conclusões chegadas e o trabalho futuro.

Capítulo 2

Revisão Bibliográfica

Neste capítulo, é apresentada a revisão bibliográfica relativa ao tema em questão.

2.1 Aplicações Móveis

O uso de dispositivos móveis tornou-se indispensável no nosso quotidiano, sendo utilizado para comunicar, aceder a produtos e serviços, bem como para entretenimento. Nos últimos anos o uso de aplicações móveis tem aumentado e, cada vez mais, as empresas tentam disponibilizar os seus serviços e produtos através de tais aplicações.

Um dos principais desafios no desenvolvimento de *software* em dispositivos móveis prende-se com o facto de se estar num cenário tecnológico altamente fragmentado e em rápida evolução. Como existe uma variedade de sistemas operativos, cada um com o seu *Software Development Kit* (SDK), *Integrated Development Environment* (IDE), linguagem de programação e *Application Programming Interface* (API), tal facto constitui um desafio para as empresas que pretendem abranger maior parte do mercado [11]. As aplicações para dispositivos móveis exigem uma quantidade razoável de personalização para serem executadas em diversas plataformas e responder ao fluxo contínuo de novos *hardwares*, versões de sistemas operativos e *browsers*. Assim, caso uma empresa pretendesse desenvolver uma aplicação para diferentes plataformas, seria necessário criar essa aplicação para cada plataforma alvo. Esta abordagem implicaria mais tempo para o ciclo de desenvolvimento da aplicação, equipas com conhecimentos específicos de cada plataforma, maior manutenção, ou seja, no cômputo geral, mais custos. Sendo assim, para o desenvolvimento rápido e economicamente viável de aplicações móveis surgiram as soluções *cross-platform*. O conceito principal de *cross-platform* é desenvolver uma aplicação uma vez e esta ser capaz de correr em várias plataformas [12].

As aplicações móveis dividem-se em três tipos:

- Aplicações nativas
- Aplicações Web
- Aplicações híbridas

Devido às diferenças na tecnologia subjacente, cada tipo de aplicação (Web, nativa e híbrida) possui vantagens e desvantagens inerentes. Um equilíbrio entre cada tipo de aplicação pode ser medido em termos de funcionalidade, desempenho, experiência do utilizador e custo total. Nas subsecções seguintes é feita a descrição de cada tipo de aplicações.

2.1.1 Aplicações Nativas

Uma aplicação nativa é uma aplicação criada especificamente para um determinado dispositivo móvel e Sistema Operativo (SO).

A intensa competição entre as empresas que dominam o mercado de SO móveis (Apple, Google e Microsoft) é responsável pelo rápido crescimento do mercado de dispositivos móveis assim como pela fragmentação dos sistemas operativos móveis [13].

Para o desenvolvimento de aplicações nativas, cada plataforma tem a sua própria infraestrutura. Assim, por exemplo, para Android as aplicações são desenvolvidas em Java e utiliza-se Android Studio como IDE ou qualquer IDE habilitado para Java. Para Microsoft Windows Phone usa-se a linguagem C#/C++ e Visual Studio como IDE [13] [14]. De notar que para o caso da plataforma iOS requer-se um computador com o sistema operativo MacOS, e usa-se a linguagem Objective C ou Swift e o IDE Xcode [15].

A criação de uma aplicação nativa requer que os desenvolvedores escrevam o código-fonte e criem recursos adicionais (imagens, áudio e ficheiros de declaração específicos do SO). O código-fonte é compilado, fazendo uso das ferramentas fornecidas pelo fornecedor do SO, para criar um executável binário que é empacotado juntamente com o resto dos recursos e pronto para distribuição. Essas ferramentas, chamadas SDK, são específicas de cada plataforma. Cada SO móvel possui as suas próprias ferramentas [16].

A Tabela 2.1 apresenta as diferentes linguagens, ferramentas, formatos e as *App Stores* dos principais SO móveis.

As aplicações nativas estão intimamente ligadas à plataforma em que correm, podendo aceder a todas as funcionalidades do dispositivo (Figura 2.1). Assim, através de API, podem aproveitar os recursos nativos do dispositivo móvel incluindo *hardware* (como GPS, câmara, gráficos, etc.) e o *software* (como contactos, galeria, gestor de ficheiros, etc.)

As aplicações nativas são rápidas, confiáveis e têm bom desempenho. Pelo facto de serem específicas a um determinado SO e para que funcionem em vários dispositivos, requerem o desenvolvimento de versões distintas para cada plataforma que se pretender alcançar [17]. Assim, o desenvolvimento, teste e manutenção de uma aplicação nativa para vários dispositivos requer um incremento na equipa de desenvolvimento e consequentemente aumento nos custos e tempo.

Tabela 2.1: Características dos sistemas operativos mais utilizados

SO	Linguagem	IDE	SDK	Formato pacote	App Store
iOS	Objective C, Swift	Xcode	iOS SDK	.app	Apple iTunes
Android	Java	Android Studio	Android SDK	.apk	Google Play
Windows Phone	C#	Visual Studio	Microsoft Windows SDK	.xap	Windows Store

2.1.2 Aplicações *Web*

Aplicações *Web* são *websites* otimizados especificamente para serem utilizados em dispositivos móveis, ou seja, têm de levar em consideração os diferentes tamanhos de ecrã, resolução e a sua filosofia de uso. O acesso é feito no *browser* do dispositivo móvel através de um *Uniform Resource Locator* (URL). Tal como num *website* tradicional, são implementadas com as tecnologias *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript [15].

Por serem baseadas no *browser*, não dependem da plataforma e dispositivo, sendo capazes de serem executadas em qualquer dispositivo compatível com a *Web* [14].

Uma aplicação *Web* normalmente é descarregada de um servidor *Web* central cada vez que é executada, embora aplicações desenvolvidas usando HTML5 também possam ser executadas no dispositivo móvel para uso *offline*.

As aplicações *Web* não exigem instalação e as atualizações processam-se de forma automática sem necessidade de aprovação nos dispositivos [11].

A principal desvantagem das aplicações *Web* é o facto do acesso às funcionalidades nativas do dispositivo (tais como sistema de notificações, GPS, lista de contactos, etc) ser limitado [14].

Como se pode ver na Figura 2.2, as aplicações *Web* não têm acesso ao microfone, câmara e a vibração. Pode-se também notar que têm um acesso limitado às outras funcionalidades nativas do telemóvel como GPS, acelerómetro, alto-falante, dados, etc.

Regista-se que o desenvolvimento e adoção do padrão HTML5 permite o acesso a componentes de *hardware* e *software* do dispositivo através de várias API e está disponível em qualquer *browser* compatível com HTML5 [11].

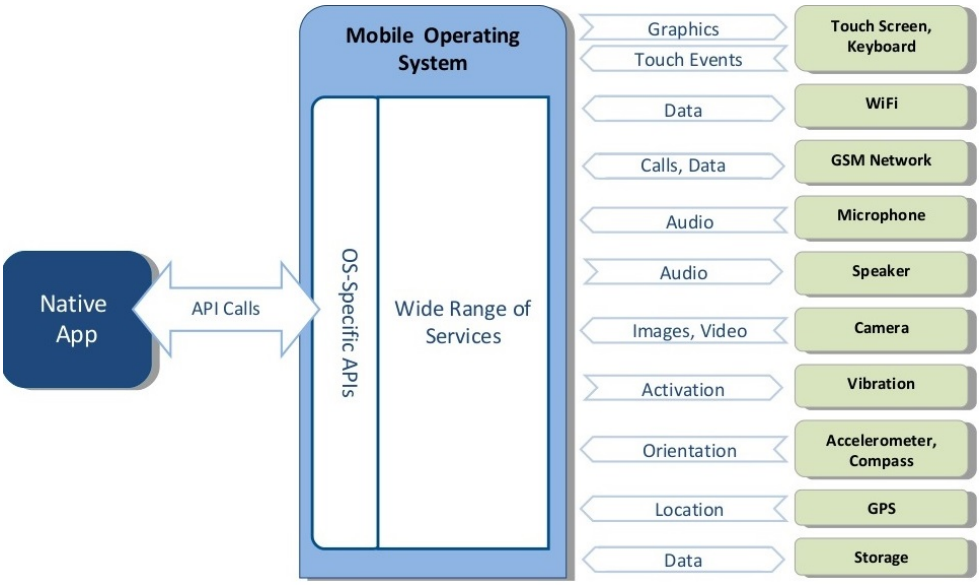

Figura 2.1: Aplicação nativa — Interação com dispositivo móvel [18]

2.1.3 Aplicações Híbridas

Uma aplicação híbrida resulta da combinação das vantagens de aplicações *Web* com as de código nativo [14]. Os desenvolvedores, aproveitam esse facto e desenvolvem uma parte significativa das suas aplicações em tecnologia multi-plataforma enquanto mantêm, quando solicitado, o acesso direto a API nativas [16]. Desta forma, parece lógico haver redução significativa no tempo e o custo de desenvolvimento, minimizando o trabalho de codificação personalizado.

As aplicações híbridas podem tirar vantagens tanto de utilizar o *browser* dos dispositivos móveis quanto da sua API. São desenvolvidas usando *frameworks* cuja finalidade é incorporar o conteúdo HTML no *Web container* nativo (*WebView* para android, *UIWebView* para iOS) e permitir que o código *Web* aceda às funcionalidades nativas do dispositivo [14] [12]. Assim, uma aplicação híbrida tem acesso aos recursos nativos do dispositivo tais como câmara, microfone, GPS (Figura 2.3).

Poucas desvantagens são encontradas nesta abordagem, que se resume ao desempenho, uma vez que são executadas no *browser*. Além disso, a interface não tem acesso à aparência nativa sendo por isso necessário usar bibliotecas de desenvolvimento específicas [14].

O utilizador não encontra diferença substancial entre uma bem projetada aplicação híbrida e nativa, pois é descarregada de um *app store*, armazenada no dispositivo móvel e carregada, como se duma aplicação nativa se tratasse. Para os desenvolvedores a diferença é substancial, pois precisam apenas de escrever alguns códigos em linguagem HTML, CSS e JavaScript e reutilizá-los nos vários dispositivos, poupando desse modo o tempo de reescrever para cada plataforma móvel a aplicação correspondente [19].

Em resumo, poder-se-á afirmar que a grande vantagem de um *framework* para aplicações híbridas é a reutilização de código que ele propicia, onde, com um único código é possível gerar a

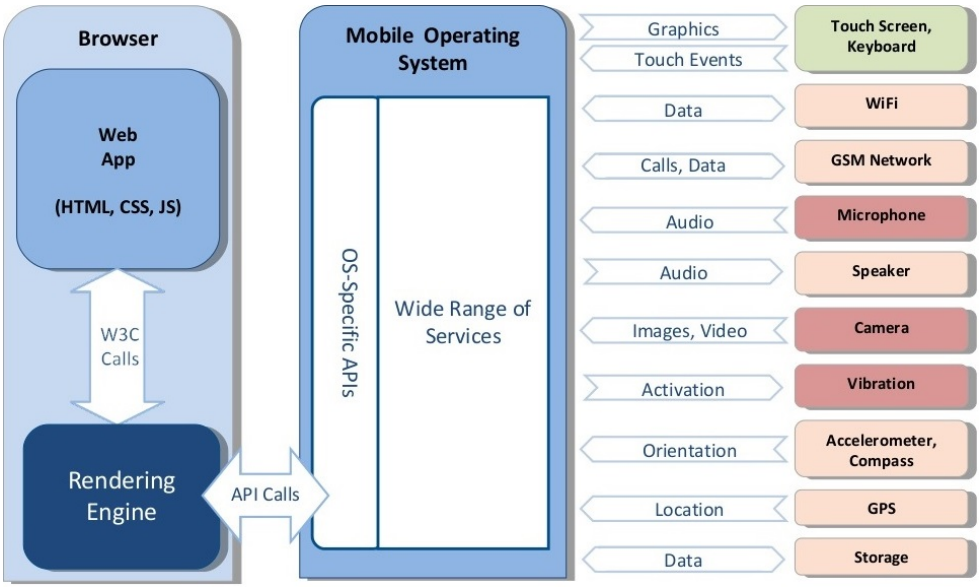

Figura 2.2: Aplicação Web — Interação com dispositivo móvel [18]

aplicação para as diversas plataformas suportadas, mantendo uma mesma interface e experiências iguais para as diversas plataformas.

2.1.4 Comparativo

A Tabela 2.2 representa o quadro comparativo das características dos três tipos de aplicações móveis.

As aplicações nativas quando comparadas com outras aplicações apresentam melhor desempenho, têm uma API completa para aceder a todas as funcionalidades do dispositivo e têm aparência nativa da UI. Porém, o desenvolvimento de aplicações nativas requer, por um lado, um alto nível de conhecimento e, por outro, as aplicações precisam de ser desenvolvidas para cada plataforma que se quer atingir, impactando desse modo no custo e tempo [12].

As aplicações *Web* são fáceis de se desenvolver quando comparadas com as aplicações nativas. Uma vez desenvolvidas, correm em diferentes plataformas através de *browsers*. A manutenção é simples pois as atualizações da aplicação e dos dados são feitos no servidor. O ponto menos conseguido das aplicações Web é que, por um lado, elas não acedem a todas as funcionalidades e, por outro, têm menos desempenho dado que a linguagem de interpretação HTML e JavaScript são analisadas e implementadas através dos *browsers* [12].

Tal como as aplicações *Web*, as aplicações híbridas são fáceis de se desenvolver. Uma vez que não requerem conhecimento de linguagem específicas das plataformas tem menos custos de desenvolvimento por não serem necessárias grandes equipas de desenvolvedores.

As desvantagens mais significativas das aplicações híbridas prendem-se com o seu desempenho quando comparadas com o das aplicações nativas. As aplicações híbridas não oferecerem aos utilizadores uma experiência nativa semelhante [12].

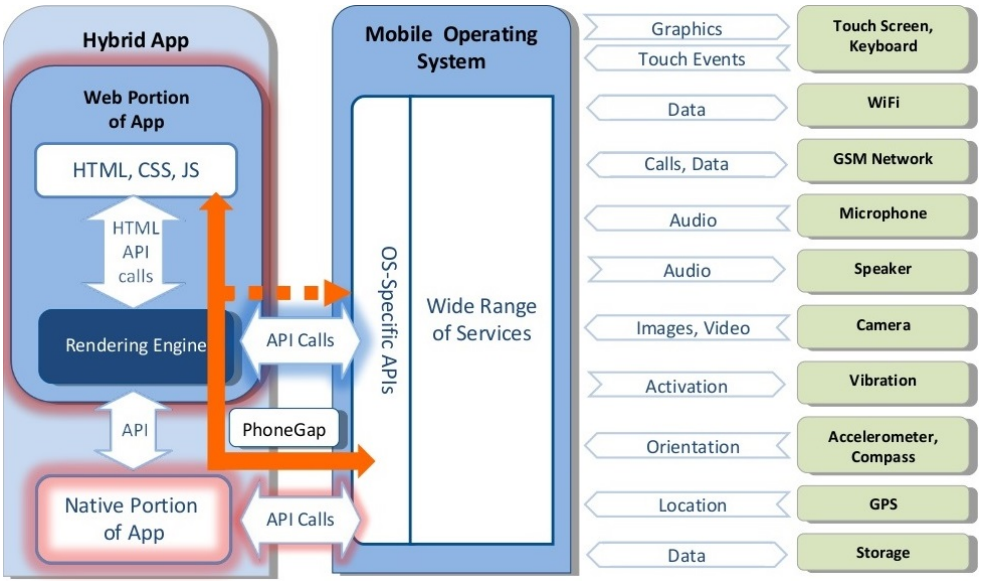

Figura 2.3: Aplicação híbrida — Interação com dispositivo móvel [18]

A decisão de qual abordagem seguir, depende muito do contexto em que se está inserido e o objetivo que se pretende atingir. Por exemplo, uma abordagem multi-plataforma (*Web* ou híbrida) pode ser a melhor solução quando se quer atingir várias plataformas e se tem como limitação o tempo e o custo do desenvolvimento.

2.2 Soluções *Cross-Platform*

Como mencionado acima neste capítulo, o conceito principal de *cross-platform* é desenvolver uma aplicação uma vez e esta ser capaz de correr em várias plataformas [12].

As soluções *cross-platform* podem seguir diferentes abordagens. Em [14], Latif et al. categoriza essas abordagens em: *Web Approach*, *Hybrid Approach*, *Interpreted Approach* e *Cross-Compiled Approach*.

2.2.1 *Web Approach*

Essa abordagem para dispositivos móveis baseia-se em *Web browsers* ou seja uma aplicação do tipo *Web* como descrito na Secção 2.1.2.

2.2.2 *Hybrid Approach*

A abordagem híbrida resulta da combinação das vantagens das tecnologias utilizadas na *Web* e as funcionalidades nativas, como descrito na Secção 2.1.3.

Tabela 2.2: Características das aplicações móveis. Adaptado de [20]

	Native	**Hybrid**	**Web**
Device Access	Full	Full	Partial
Speed	Native speed	Very fast	Fast
App Development Cost	Expensive	Reasonable	Reasonable
AppStore	Yes	Yes	No
Approval Process	Mandatory	Low overhead	None
Quality of UX	Excellent	Not as good as native apps	Very good
Quality of Apps	High	Medium to low	Low
Potential Users	Limited to a particular mobile platform	Large as it reaches to users of different plataforms	Maximum including smartphones, tablets and other feature phones
Access Device Specific Features	High	Medium	Low
Develomenpt Language	Objective-C, Java, C, C++, C#	HTML, CSS, JavaScript	HTML, CSS, JavaScript

2.2.3 *Interpreted Approach*

A *Interpreted Approach* usa uma linguagem de programação comum como por exemplo JavaScript, para escrever o código da *user interface* e gerar o equivalente para o componente nativo de cada plataforma. A obtenção dos recursos nativos é conseguida através duma camada abstrata que, nas diferentes plataformas, interpreta o código em tempo de execução e deste modo tem acesso às API nativas. Assim como nas outras abordagens, a *interpreted approach* apresenta vantagens e desvantagens. A sua grande vantagem reside no uso de interfaces nativas e a desvantagem é a sua completa dependência do ambiente de desenvolvimento da aplicação [14].

2.2.4 *Cross-Compiled Approach*

A abordagem *Cross-Compiled Approach* permite aos desenvolvedores, através do uso de qualquer linguagem comum, escreverem os códigos que, por meio de compiladores, são transformados em código nativo específico. A vantagem desta abordagem é poder igualar o desempenho nativo e em simultâneo fazer a entrega de todos os recursos das aplicações nativas e componentes nativos da interface. Por outro lado existem algumas limitações no uso de alguns recursos, específicos das funções variáveis de plataforma a plataforma, como por exemplo os serviços de localização gráfica e acesso à câmara [14].

2.3 *Frameworks*

Nesta secção é apresentada uma breve descrição de alguns *frameworks* para desenvolvimento de aplicações multi-plataforma.

2.3.1 Xamarin

Xamarin é um *framework* da Microsoft que permite criar aplicações nativas para Android, iOS e Windows e segue a abordagem *Cross-Compiled*. Utiliza como linguagem o C# e permite que o código seja partilhado com várias plataformas [1]. As aplicações Xamarin são construidas com *user interface* (UI) nativas, aproveitam a aceleração de hardware específica da plataforma e são compiladas para o desempenho nativo. Têm acesso a todas as funcionalidade da plataforma e dispositivo, incluindo recursos específicos, como iBeacons e Android Fragments [21]. No entanto requerem escrita de um código específico para beneficiar de características específicas da plataforma [14].

Usam a mesma linguagem, API e estruturas de dados para partilhar uma média de 75% do código da aplicação em todas as plataformas de desenvolvimento para dispositivos móveis.

2.3.2 Appcelerator Titanium

O Titanium é um *framework open source* da Appcelerator para o desenvolvimento de aplicações nativas a partir de um único código baseado no JavaScript e segue a abordagem *Interpreted*. A Appcelerator fornece junto com o Titanium duas ferramentas de grande ajuda no desenvolvimento de aplicações: o Titanium Studio e o Appcelerator Alloy. Titanium Studio é um IDE baseado em Eclipse que procura facilitar a configuração do ambiente de desenvolvimento para as diferentes plataformas. O Appcelerator Alloy, aplicado ao Titanium, permite desenvolver aplicações seguindo o padrão *Model View Controller* (MVC) e *Cloud Services*. O Titanium tem acesso a API nativas dos dispositivos e adapta a UI à convenção de cada SO. Cerca de 70-80% do código pode ser reutilizado entre aplicações [2].

2.3.3 PhoneGap

PhoneGap é uma ferramenta de desenvolvimento de aplicações de código aberto gerida pela Adobe. Permite que desenvolvedores e empresas criem aplicações livres, comerciais e de código aberto [3] . O ambiente de desenvolvimento é multi-plataforma e permite a criação de aplicações para Android, Bada, BlackBerry, iOS, Symbian, webOS e Windows Phone OS. Utiliza HTML, HTML5, CSS, CSS3 e JavaScript [22].

O PhoneGap é, essencialmente, um *"wrapper"* que permite que os desenvolvedores incluam aplicações escritas em linguagens de programação conhecidas em aplicações nativas [22].

[1] Xamarin em https://www.xamarin.com/platform.
[2] Appcelerator em https://www.appcelerator.com/mobile-app-development-products/.
[3] PhoneGap em http://phonegap.com/.

As aplicações PhoneGap são híbridas e permitem a utilização de API nativas. A renderização no dispositivo móvel é feita via *webview* [17]. O PhoneGap não fornece um IDE para desenvolver aplicações, tendo os desenvolvedores que escrever o código-fonte em diferentes IDE (por exemplo, Eclipse para Android, XCode para iOS, etc.). Esta abordagem não permite que se tenha um ambiente de desenvolvimento centralizado. Portanto o esforço necessário para compilar o código-fonte e produzir a aplicação executável (produto final) é alto [22]. De referir que quando a Adobe adquiriu o PhoneGap, doou o código base a Apache Software Foundation, renomeando o PhoneGap em código aberto Cordova [23].

2.3.4 Framework7

O Framework7 é um *framework open source* para o desenvolvimento de aplicações móveis híbridas ou aplicações *Web* com aparência nativa. Não é compatível com todas as plataformas, estando focado apenas para iOS e Android. Permite a criação de UI de forma simples utilizando aspetos nativos do SO [4]

2.3.5 Ionic

O Ionic é um *framework open source* usado para desenvolver aplicações móveis híbridas e é construído em cima de AngularJS e Apache Cordova. Fornece ferramentas e serviços para desenvolver aplicações móveis híbridas usando tecnologias Web como CSS, HTML5 e JavaScript. Através de *plugins* do Cordova, as aplicações têm acesso aos API nativos dos dispositivos móveis [5].

2.4 Plataforma de *e-Science* WindsP

WindsP é a plataforma *Web* desenvolvida pela Universidade do Porto para apoio à experiência de campo do projeto NEWA que decorre em Perdigão. Suporta o trabalho dos gestores da campanha Perdigão e investigadores, bem como as instituições participantes, desde o planeamento, preparação, disseminação dos resultados relevantes e os respetivos arquivos. O WindsP permite aos investigadores pesquisar e utilizar os *datasets* resultantes da campanha. O WindsP fornece, ainda, a localização das estações e equipamentos usados na recolha de dados experimentais, listagem de eventos relevantes e documentação produzida [8].

2.4.1 Funcionalidades da Plataforma WindsP

WindsP inclui as seguintes funcionalidades:

- Pesquisas de intenção de colaboração: Ao construir uma estrutura de categorização, recolhe a intenção de colaboração dos parceiros bem como informação sobre equipamentos.

[4] Framework7 em https://framework7.io/.
[5] Ionic em http://www.ionicframework.com.

- Informação geral da finalidade da experiência: informações gerais que serão úteis para todos os participantes da experiência, tais como, localização, contacto, datas, topografia e mapas de rugosidade.
- Resumo das intenções de colaboração por experiência: resumo de uma página de todos os recursos para uma experiência (equipamentos, modelos de fluxo de vento e pessoal).
- Instalação e equipamentos de estações de campo: uma funcionalidade que permite ao gestor da campanha criar estações e equipar as estações torre com o equipamento necessário.
- Inventário de equipamentos: a base de dados de todos os equipamentos por categoria e fabricante, com folhas de dados e manuais.
- *Logbook*: registo de eventos de campanha de diferentes categorias e visibilidades diversas (*Issue, Schedule, Installation, Social*, etc).
- Gestor de documentos: gestão de uma "Dropbox" com pastas e arquivos relacionados com a campanha.
- Cronogramas: gestão de pessoal e equipamentos da campanha.
- Funcionalidades de gestão administrativa: gestão de toda a plataforma (contas de utilizadores, equipamentos, eventos do *logbook*, documentos, criação de dados dos modelos de fluxo de vento).
- Pesquisa de *datasets*: *datasets* produzidos estão em grande número e em formatos diferentes; os investigadores podem explorar dados no arquivo da Universidade do Porto a partir do WindsP e filtrar ou descarregar os ficheiros desejados usando o *Thredds data server* (TDS).

2.4.2 Ilustrações das Funcionalidades do WindsP

As figuras abaixo, ilustram algumas funcionalidades da plataforma WindsP.

- Figura 2.4 — *Home* do WindsP contém informações gerais sobre a campanha, localização e um contacto de referência.
- Figura 2.5 — *Dashboard* do gestor de campanha. Ele é responsável pela gestão das torres e estações sensoriais.
- Figura 2.6 — Área de documentação do WindsP. O objetivo é reunir toda a documentação sobre a campanha num único lugar de modo a simplificar a consulta e o acesso dos participantes da campanha.
- Figura 2.7 — Mapa com a localização das estações. As estações podem ser filtradas por categorias. Os participantes podem exportar a informação sobre as estações para CSV ou KML.

- Figura 2.8 — Informações de uma estação. Cada estação tem uma página com informações relevantes, como coordenadas, equipamentos instalados, datas importantes, etc.
- Figura 2.9 — *Logbook* do WindsP. O objetivo do *logbook* é registar atividades do dia-a-dia da campanha. Os participantes podem criar eventos e associar documentos e fotos.
- Figura 2.10 — *Personnel Timeline*. Os utilizadores podem ver quando um participante da experiência está no terreno. O cronograma pode ser agrupado por dias, semanas ou meses. Quando agrupado por dia, é possível ver o número de pessoas que estão no terreno.

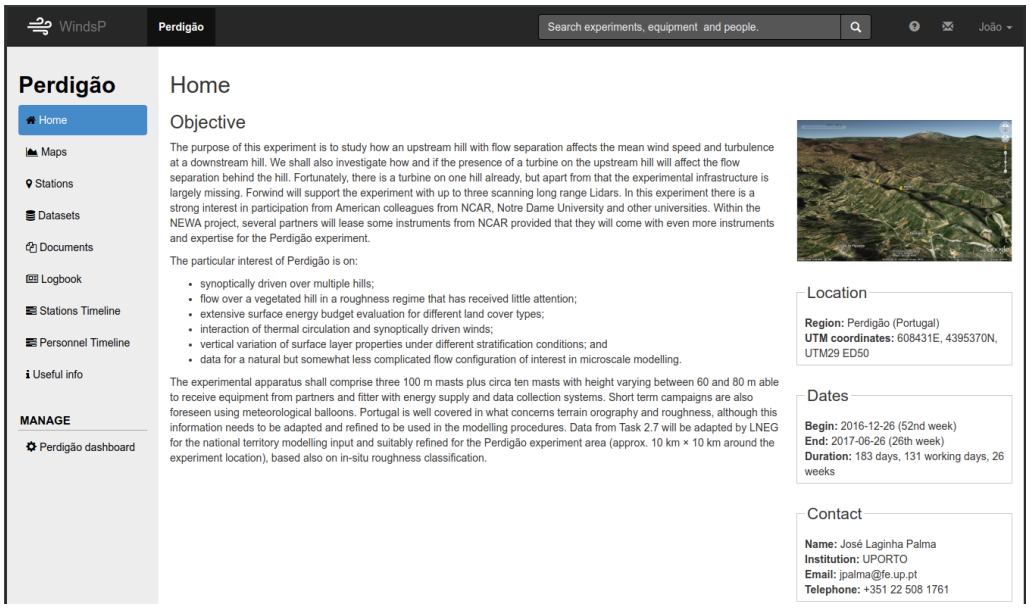

Figura 2.4: *Home*

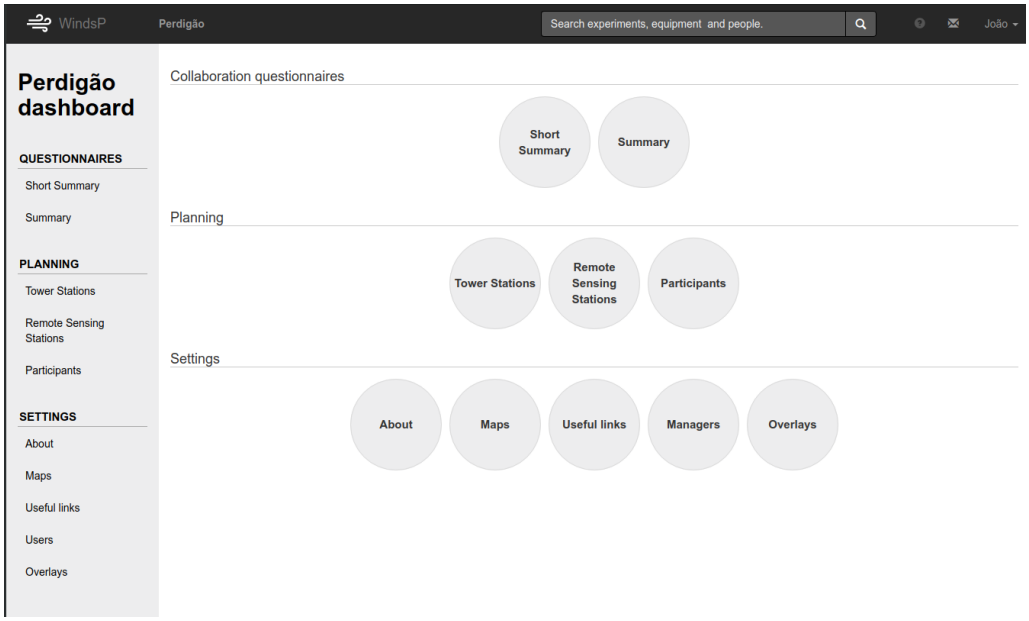

Figura 2.5: *Dashboard*

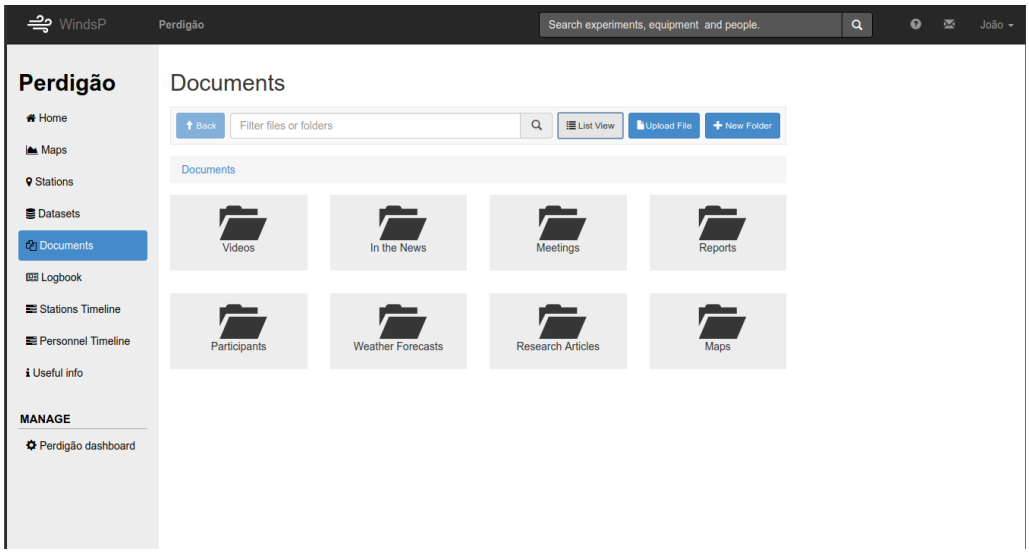

Figura 2.6: *Documents*

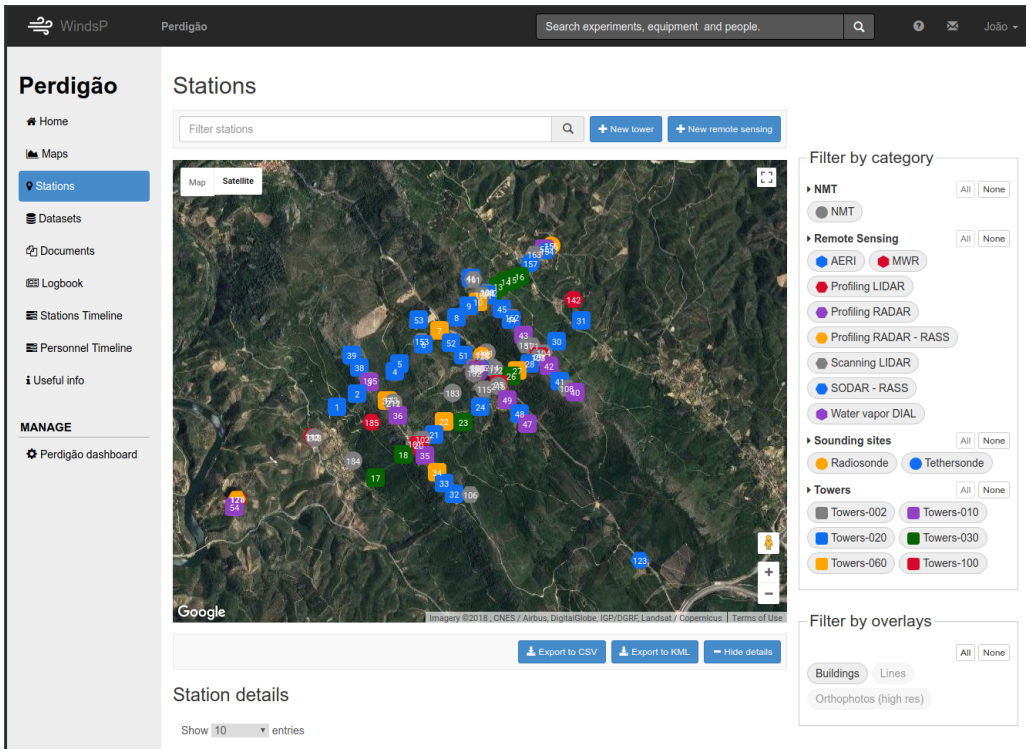

Figura 2.7: *Stations map*

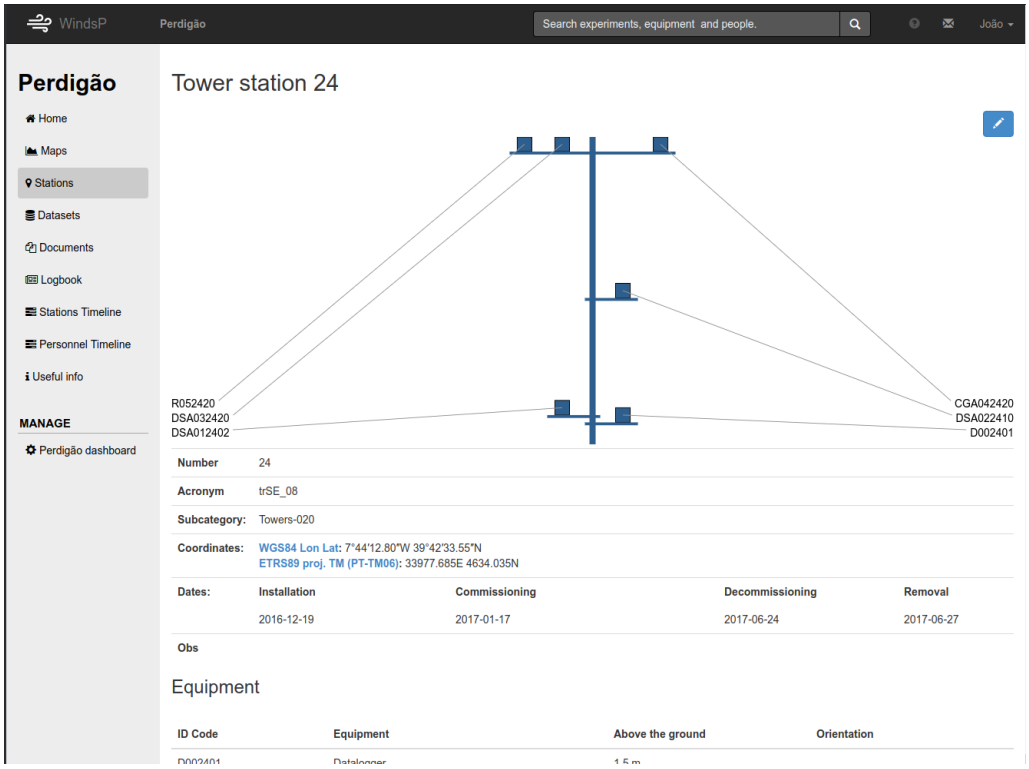

Figura 2.8: *Station info*

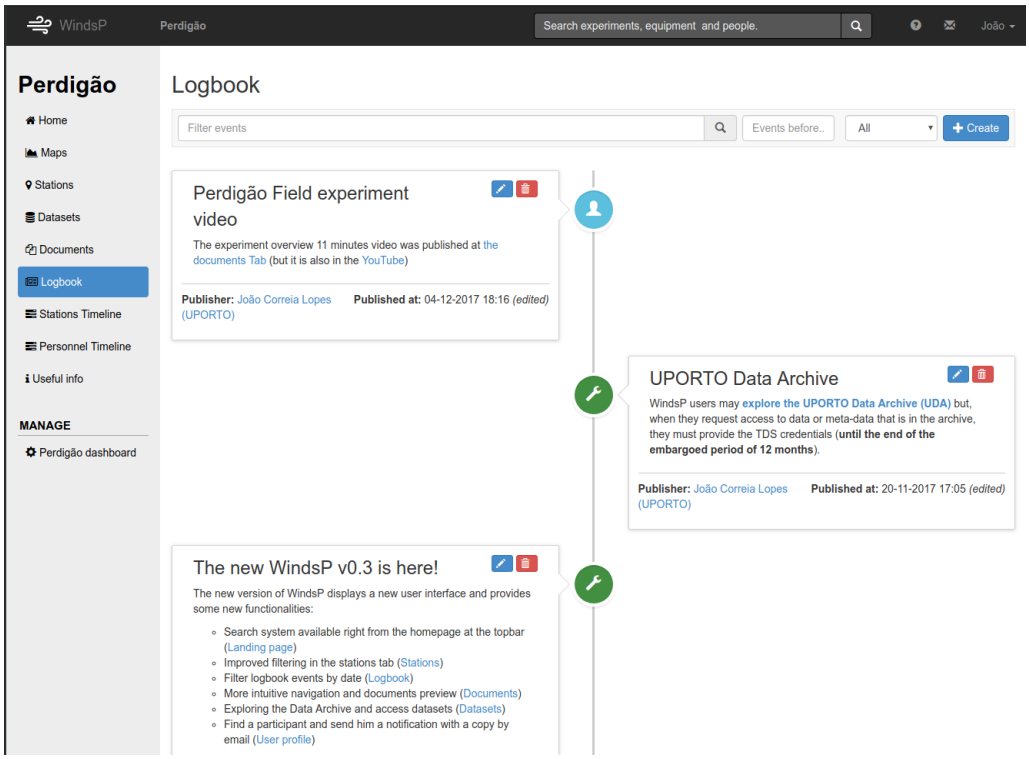

Figura 2.9: *Logbook*

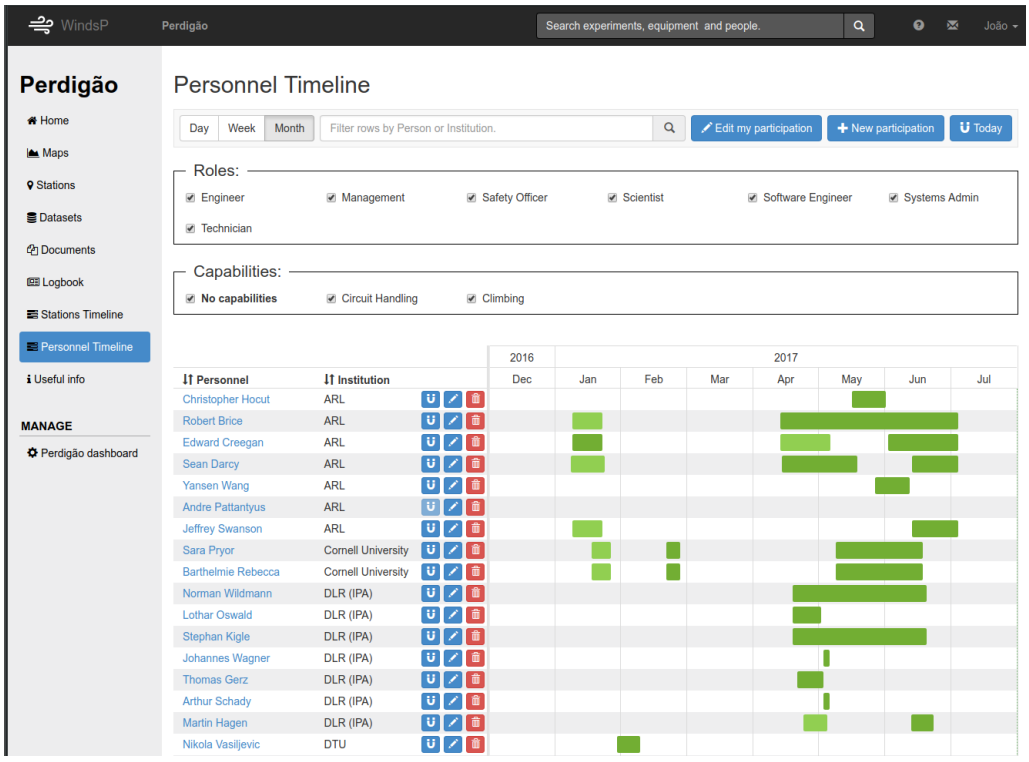

Figura 2.10: *Personnel timeline*

2.5 Resumo

Neste capítulo foram abordados os tipos de aplicações móveis e os *frameworks* de desenvolvimento. Na última secção foi também abordada a plataforma *Web* desenvolvida pela Universidade do Porto de apoio à experiência de campo do projeto NEWA que decorreu em Perdigão.

Capítulo 3

Análise de Requisitos

Neste capítulo apresenta-se a análise dos requisitos para a aplicação desenvolvida. Os requisitos de *software* são tradicionalmente classificados como requisitos funcionais e não funcionais.

3.1 Introdução

Como referido na Secção 1.2, na campanha Perdigão estão instaladas dezenas de estações de recolha de dados, sendo difícil saber que equipamentos estão instalados em cada estação.

Deste modo, queremos oferecer aos investigadores que estão no terreno uma ferramenta que lhes permita localizar as estações que se encontram à sua volta, verificar os equipamentos instalados em cada estação e, também, reportar eventos que estejam a decorrer.

A reunião tida com alguns dos *stakeholders* da campanha Perdigão deu origem aos requisitos funcionais descritos na Secção 3.2. Esses requisitos funcionais são apresentados na forma de *user stories*.

3.2 Requisitos Funcionais

Requisitos funcionais descrevem o que um sistema deve fazer, dependem do tipo de *software* a ser desenvolvido, dos utilizadores alvos e da abordagem da organização que desenvolve o *software* [24].

3.2.1 Atores

"Um ator é o conceito que representa, em geral, um papel que um utilizador desempenha relativamente ao sistema em análise. Todavia, um ator não é necessariamente um papel de um utilizador; pode corresponder a um papel desempenhado por um outro sistema informático, por um equipamento hardware especializado, ou pela simples passagem de tempo. O conjunto total de atores de todos os casos de utilização reflete todos os elementos que interatuam com o sistema" [25].

Na aplicação consideram-se dois atores:

- Visitante: é um utilizador não autenticado. Pode introduzir os dados de autenticação para fazer *login* na aplicação.
- Investigador: é um utilizador autenticado que pode aceder a todas as funcionalidades da aplicação.

NB: O registo dos utilizadores é feito na plataforma WindsP.

3.2.2 *User Stories*

User Story é uma definição considerada de alto-nível de um requisito de sistema, contendo informação suficiente que permite aos desenvolvedores estimar o esforço necessário à sua implementação. É contada na perspetiva do utilizador ou cliente do sistema [26], [27].

Na Tabela 3.1 são listados os *Users Stories* da aplicação.

Tabela 3.1: *User stories*

Id	**Nome**	**Descrição**
US01	Fazer *login*	Como *Visitante* quero autenticar-me no sistema para ter acesso a informação restrita
US02	Ver informação da campanha	Como *Investigador* quero ver o mapa das estações para saber a localização de cada estação.
US04	Ver *logbook*	Como *Investigador* quero ver o conteúdo das categorias públicas de eventos do *logbook* da campanha, para que conheça os eventos relacionados às informações que posso ver.
US03	Publicar eventos	Como *Investigador* quero publicar eventos no *logbook* relativos à campanha, para conhecimento dos outros participantes.
US04	Editar eventos	Como *Investigador* quero editar eventos que criei, a fim de atualizá-los.
US05	Remover eventos	Como *Investigador* quero remover eventos que criei para que possa corrigir erros.
US06	Publicar fotografia eventos	Como *Investigador* quero publicar fotografias, para conhecimento dos outros participantes.
US07	Filtrar eventos por texto	Como *Investigador* quero filtrar eventos usando palavras-chave para ter acesso a informações do meu interesse disponíveis.
US08	Filtrar eventos por data	Como *Investigador* quero filtrar eventos por data para ter acesso a informações disponíveis num intervalo de tempo do meu interesse.

US09	Filtrar eventos por categorias	Como *Investigador* quero filtrar eventos por categoria para ter acesso a informações disponíveis e do meu interesse.
US11	Publicar localização nos eventos	Como *Investigador* quero publicar a minha localização na altura da publicação do evento para conhecimento de outros participantes.
US12	Ver perfil	Como *Investigador* quero ver o perfil do autor de um determinado evento, para que possa ter acesso a informações básicas do mesmo.
US13	Ver mapa das estações	Como *Investigador* quero ver o mapa da estações para saber a localização de cada estação.
US14	Filtrar estações por categoria	Como *Investigador* quero filtrar as estações por categoria para que possa ver no mapa somente as do meu interesse.
US15	Filtrar estações dado um raio	Como *Investigador* quero filtrar as estações dado um raio relativo à minha localização para que possa ver no mapa as estações à minha volta.
US16	Ver informação das estações	Como *Investigador* quero ver informações específicas de cada estação para que eu possa saber a informação detalhada dos equipamentos instalados.
US17	Seguir localização	Como *Investigador* quero utilizar a geolocalização para saber a minha localização relativamente às estações à medida que me desloco.
US18	Aceder câmara	Como *Investigador* quero aceder à câmara fotográfica para registo de acontecimentos a fim de serem publicados.
US19	Aceder galeria	Como *Investigador* quero ter acesso à galeria de fotografia para publicar nos eventos ou nas estações.
US20	Publicar fotografia de estações	Como *Investigador* quero publicar fotografias de estações a fim de detalhar e complementar informações das mesmas.
US21	Remover fotografia de estações	Como *Investigador* quero remover fotografias de estações para corrigir eventuais erros.

3.3 Requisitos Não Funcionais

Ao contrário dos requisitos funcionais apresentados na Secção 3.2, que estão relacionados com os serviços oferecidos pelo sistema aos utilizadores, os requisitos não funcionais estão relacionados com as propriedades ou características que o sistema deve cumprir ou restrições que deve

respeitar, tais como confiabilidade, desempenho, tempo de resposta, segurança, etc., assegurando desse modo a qualidade do sistema [24], [28].

Na Tabela 3.2 são listados os requisitos não funcionais do sistema.

Tabela 3.2: Requisitos não funcionais

Id.	**Nome**	**Descrição**
RNF01	Disponibilidade	A aplicação deve estar disponível a maior parte do tempo.
RNF02	Desempenho	A aplicação deve ter um tempo de resposta que não degrada a experiência do utilizador.
RNF03	Manutenção	A aplicação deve ser projetada e implementada para facilitar o isolamento e correção de defeitos e suas causas.
RNF04	Usabilidade	A aplicação deve ser simples e fácil de usar, com botões de fácil interação com os dedos e formulários simples de fácil interação para o utilizador.
RNF05	*Open source*	A aplicação deve ser implementada usando tecnologias de código aberto.
RNF06	Multi-plataforma	A aplicação deve ser compatível e acessível através de um *smartphone* ou *tablet* com o sistema operativo móvel Android ou iOS e com ecrãs de diferentes tamanhos e resoluções.
RNF07	Segurança	A utilização da aplicação só é possível após a autenticação.
RNF08	Confiabilidade	A aplicação deve executar consistentemente de acordo com suas especificações, minimizando os erros técnicos.
RNF09	Funcionamento *online*	A aplicação necessita de ligação à Internet para fazer o *login*, carregar e atualizar dados.
RNF10	Mensagens de erro	A aplicação deve apresentar mensagens de erros durante a autenticação e invalidação.

3.4 Resumo

Neste capítulo foram descritos os requisitos da aplicação, ou seja, o que o sistema deve fazer. Também foram abordados os aspetos relacionados com as propriedades ou características que o sistema deve cumprir ou restrições que deve respeitar, tais como confiabilidade, desempenho, segurança, etc. Os requisitos definidos vão permitir, como se segue no Capítulo 4, escolher as tecnologias a utilizar e desenhar a arquitetura do sistema.

Capítulo 4

Arquitetura

Neste capítulo do desenho da arquitetura da aplicação aborda-se, de entre as várias soluções tecnológicas, a solução que mais se adequa ao problema levantado.

4.1 Introdução

Após a análise dos diferentes tipos de aplicações móveis (nativa, *Web* e híbrida) e, comparando as vantagens/desvantagens de umas e de outras e tendo em conta que os investigadores utilizam dispositivos móveis de diferentes plataformas, consideramos que a abordagem híbrida é a melhor solução para o caso em apreço. Por outro lado e reforçando a nossa opção, temos o facto dos investigadores da campanha Perdigão usarem a plataforma *Web* WindsP e, por isso, faz todo o sentido utilizar o *back-end* deste e aproveitar diretivas criadas.

4.2 Tecnologias Escolhidas

Para a implementação da aplicação móvel foi escolhido o Ionic *framework*. Essa escolha prende-se com o facto de ser uma plataforma com uma grande comunidade *online* e bom suporte técnico. Baseia-se em tecnologia *Web*, o que permite uma portabilidade do código desenvolvido, ou seja, no final do desenvolvimento poder exportar essa aplicação para Android e iOS. Outra razão ser uma ferramenta e tecnologia gratuitas ou de código aberto (*open-source*).

4.2.1 Ionic

Como referido na Secção 2.3.5, Ionic é um *framework open source* utilizado para desenvolver aplicações híbridas rápidas. Portanto, uma aplicação Ionic não é executada num *browser* normal do dispositivo como o Chrome ou Safari, mas sim num *browser* de baixo nível, *webview* (*UIWebView* para iOS e *WebView* para o Android). Utiliza tecnologias Web modernas como HTML5, CSS3 e JavaScript e é construída sobre os *frameworks* Cordova e AngularJS.

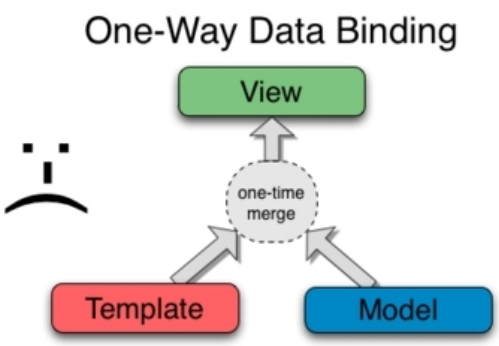

Figura 4.1: *One-way Data Binding* [30]

4.2.2 AngularJS

O AngularJS é um *framework open source* baseado em JavaScript mantido pela Google para desenvolvimento de aplicações *Web* dinâmicas. Foi criado tendo em mente aplicações *Create-Read-Update-Delete* (CRUD) que representam a maioria das *web apps*. É usado em projetos de *Single Page Application* (SPA).

O AngularJS dispõe de funcionalidades importantes, tais como extensão do HTML, através de diretivas (*directives*), *data-binding*, *dependency injection*, etc. Uma das vantagens mais importantes do AngularJS é a sua capacidade de organizar os códigos em pequenos módulos (*modules*), o que permite uma estrutura bem definida, facilitando aos desenvolvedores manter o código simples e reutilizar o mesmo onde for necessário [29].

4.2.2.1 *Data binding*

Data binding em AngularJS é a sincronização automática de dados entre os componentes *model* e o *view*. O *view* é a projeção do *model*. Quando o *model* muda, o *view* reflete a mudança e vice-versa [30].

O *Data binding* é dividido em dois tipos:

- *One-Way Data Binding* para sistemas clássicos de *template*;
- *Two-Way Data Binding* para *templates* do AngularJS.

O ***One-Way Data Binding*** é utilizado na maior parte dos sistemas clássicos de *template*. A ligação dos dados faz-se numa única direção, juntando-se os componentes *template* e *model* no *view*. Após a fusão, as alterações no *model* ou em secções do *view* não são automaticamente refletidas ou seja, o *view* não se atualiza quando o *model* mudar e vice-versa. É necessário escrever código para que a atualização se faça. Não se trata de um processo de sincronização, mas de processamento de dados numa só direção como se pode ver na Figura 4.1 [30].

O ***Two-Way Data Binding*** trabalha de forma diferente, ou seja, primeiro o *template* (que é o HTML não compilado, juntamente com *markup* ou *directives* adicionais) é compilado no *browser*.

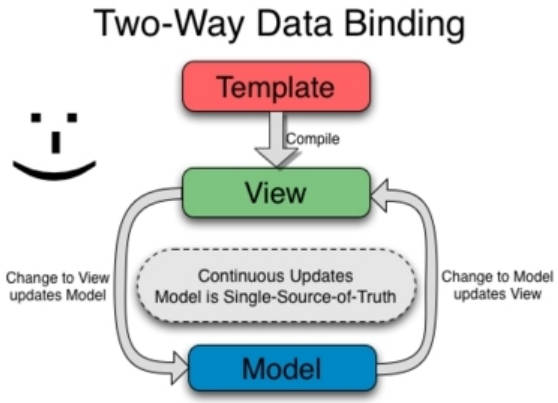

Figura 4.2: *Two-way Data Binding* [30]

A compilação produz um *view* em "direto", o que significa que qualquer alteração no *view* é refletida no *model* e qualquer alteração no *model* é propagado para o *view* (Figura 4.2). Porque o *view* é apenas uma projeção do *model*, o *controller* está completamente separado do *view* e não tem conhecimento do mesmo [30].

4.2.2.2 Padrão de arquitetura do AngularJS

O padrão de arquitetura do AngularJS tem sido alvo de muitas discussões, em que muitos se referem como sendo *Model-View-Controller* (MVC) e outros como *Model-View-ViewModel* (MVVM). Isto porque o AngularJS não segue o MVC tradicional, mas algo próximo ao MVVM. Contudo, os autores decidiram apelidar o padrão como sendo *Model-View-Whathever* (MVW). Independentemente do nome que se queira atribuir ao padrão, o importante é que o AngularJS faz a separação entre as camadas da aplicação de forma a permitir modularidade, flexibilidade e testabilidade [31].

O padrão de arquitetura MVC tem como finalidade dividir a aplicação *Web* numa estrutura suscetível de ser melhor gerida. Assim, esta arquitetura divide-se em três importantes elementos: o *model*, o *view* e o *controller* [32].

- ***Model*** — é a estrutura de dados atrás de uma parte específica da aplicação. Os dados que podem ser estáticos ou obtidos de forma dinâmica usando *JavaScript Object Notation* (JSON). Compreende um objecto JavaScript chamado *Scope*.
- ***View*** — é a secção da aplicação visível aos utilizadores ou seja a camada de interação. É composta por elementos HTML e diretivas.
- ***Controller*** — controla toda a interação entre *view* e *model*, ou seja controla toda a lógica da aplicação.

O padrão MVVM resulta do MVC mas, ao contrário deste, suporta *two way data binding* entre o *view* e o *model*. Divide-se em *Model*, *View* e *ViewModel* (Figura 4.3).

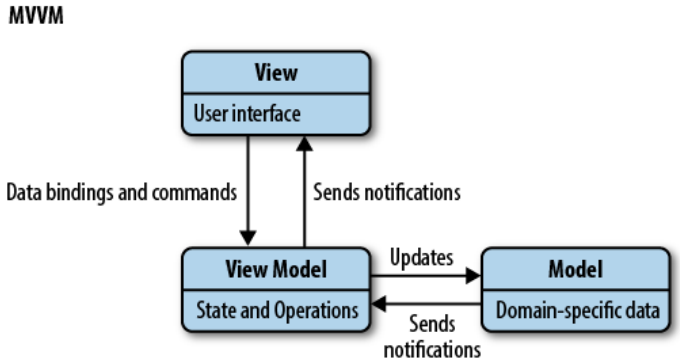

Figura 4.3: Padrão MVVM [32]

- ***Model*** — refere-se à camada de dados.
- ***View*** — é responsável pela estrutura, *layout* e aparência do que o utilizador vê. O *view* obtém os dados necessários para a apresentar, do *ViewModel* através de *data bindings* ou invocando métodos do *ViewModel*.
- ***ViewModel*** — atua entre o *view* e *model* como um intermediário e é responsável pela a lógica do *view*. Desempenha o papel do *controller* sendo que a diferença encontra-se no facto de permitir *two way data binding*.

4.2.3 Apache Cordova

O Cordova é um *framework open-source* de desenvolvimento de aplicações móveis que permite o uso de tecnologias *Web* como HTML5, CSS3 e JavaScript para o desenvolvimento multi-plataforma de aplicações.

O Cordova empacota o código não nativo juntamente com ficheiros de configuração do projeto a desenvolver dentro de uma app nativa (Figura 4.4). A aplicação é executada num *webview* que é responsável pela renderização da app. Através de *plugins*, o Cordova permite o acesso às funcionalidades nativas do dispositivo como câmara, contactos, GPS, etc. De notar que, caso o desenvolvedor tenha algum domínio das linguagens nativas da plataforma a que a aplicação se destina, é possível criar *plugins* customizados [33].

4.3 Arquitetura do Sistema

A aplicação móvel requer comunicação com um servidor de modo a aceder e armazenar informação nas bases de dados. O processo de comunicação segue o modelo Cliente-Servidor. O cliente, que é o foco deste trabalho, consiste numa aplicação móvel híbrida para *smartphone* que usa uma API REST. Os clientes solicitam e recebem serviços do servidor (Figura 4.5).

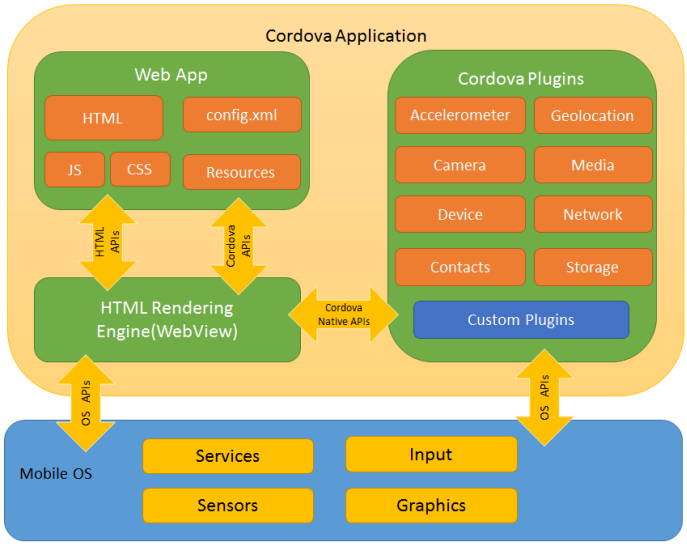

Figura 4.4: Arquitetura do Cordova [33]

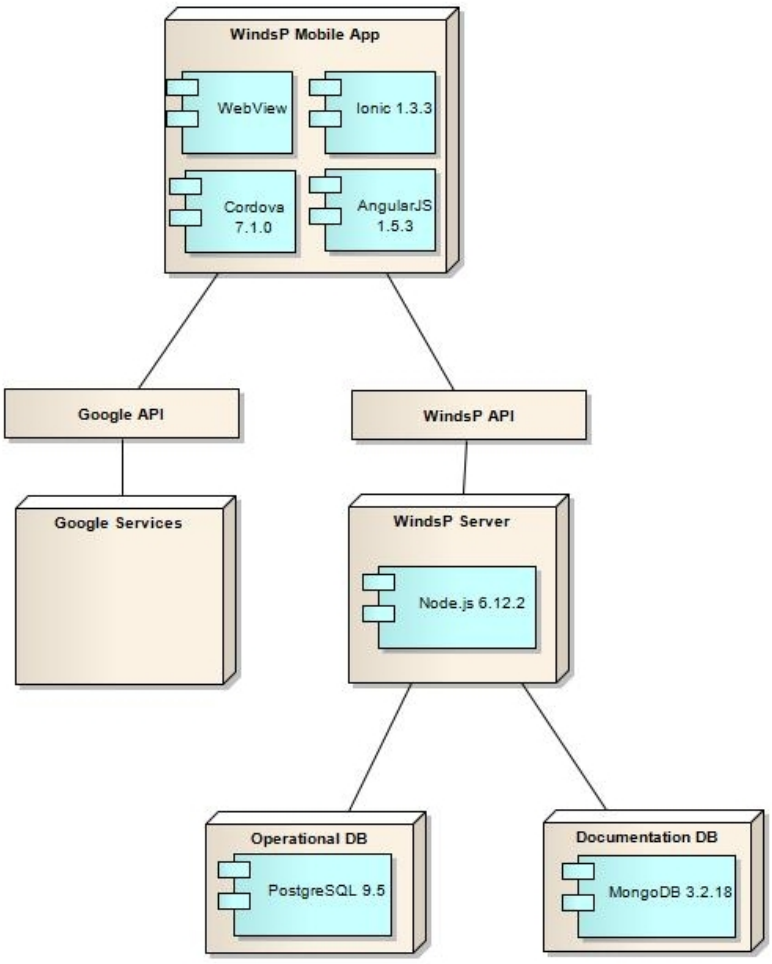

Figura 4.5: Diagrama de arquitetura

4.3.1 Servidor

O servidor da plataforma *e-Science* WindsP foi utilizado como *back-end* da aplicação móvel. Foi desenvolvido pela equipa responsável pela implementação do WindsP, utilizando o Node.js como *framework*. A comunicação com WindsP é feita através de uma Restful API. A API REST foi criada utilizando a Express.js, que é um *framework open-source* de desenvolvimento para a *Web*, escrita em JavaScript. Para a representação e troca de dados é usado o formato *JavaScript Object* (JSON).

Quanto ao armazenamento de dados, o servidor utiliza uma base de dados relacional PostgreSQL para guardar toda a informação de gestão da campanha e uma base de dados não relacional MongoDB para guardar todos os documentos (PDF, JPG, etc).

4.3.2 Aplicação Móvel

O foco do trabalho é o desenvolvimento de uma aplicação móvel. Como foi referido anteriormente, a componente móvel que foi desenvolvida corresponde a uma aplicação móvel híbrida desenvolvida com o *framework* Ionic. A aplicação comunica com o servidor através de uma API REST por forma a aceder à base de dados.

4.4 Estrutura de uma Aplicação Ionic

O Ionic é, no fundo, uma aplicação Angular que pode ser dividida em cinco partes: *Views*, *Controllers*, *Data* (*services*, *factories*), *App configuation* e *Diretives*, em que o *view*, *data* e *controllers* aproximam-se do *view*, *model* e *viewModel* do padrão MVVM [34].

- ***Views*** — é camada responsável pela apresentação. São muitas vezes referidos como *templates* em Ionic, isto porque em AngularJs são assim designados. Por omissão, o Ionic organiza as *views* numa pasta chamada *template*, onde cada *view* é um ficheiro HTML.

- ***Controllers*** — é a camada responsável por controlar todo o fluxo de dados e lógica da aplicação. Usa o *view* como *template* que apresenta a interface ao utilizador e faz chamadas às camadas de dados (*factories*/*services*) de forma a obter dados necessários à criação de um *view*. Estes dados são atribuídos ao objeto ***$scope***. O *Scope* tira partido do *two-way data binding*, permitindo que o *template* se atualize automaticamente quando os valores do ***$scope*** são alterados e, ainda, que os campos de *input* do *template* vinculados ao *scope*, ao serem alterados pelo utilizador, estejam disponíveis no controlador.

- ***Data (services/factories)*** — é a camada que disponibiliza dados solicitados pelo controlador, que provêm, geralmente, de um *back-end* externo ou *Web service*. As chamadas ao *back-end* são feitas normalmente usando o serviço ***$http***.

- ***App configuration*** — é a camada que contém os métodos ***run*** e ***config*** do AngularJS. Nessa camada são definidas as rotas ou estados da aplicação e os controladores são ligados aos seus *templates* correspondentes.

- ***Directives*** — essa camada contém *directives* personalizadas que a aplicação pode utilizar. As *directives* podem ser vistas como sendo atributos ou classes que desencadeiam comportamentos específicos num elemento de uma página HTML.

4.5 Resumo

O objetivo deste capítulo prende-se com a escolha de tecnologias e desenho da arquitetura do sistema que melhor servem os propósitos do projeto. De entre as várias abordagens para desenvolvimento de aplicações móveis, a híbrida foi a selecionada e para sua implementação escolhida o Ionic *framework*.

Capítulo 5

Implementação

Neste capítulo é caracterizado o ambiente de desenvolvimento e são apresentados os aspetos técnicos de detalhe relativos à conceção da aplicação. São ainda abordados aspetos relativos à instalação da aplicação e organização do código.

5.1 Ambiente de Desenvolvimento

Nesta secção é apresentada o ambiente de desenvolvimento utilizado durante a implementação.

- **Gestão do código:** Git
- **Sistema Operativo:** Ubuntu 17.10
- **IDE:** Visual Studio Code v1.19.1
- **Linguagem de programação:** JavaScript, HTML5, CSS
- ***Frameworks:*** Ionic Framework: ionic1 v1.3.3; Cordova: v7.1.0; Angular.JS: v1.5.3; Node.js: v6.11.4;
- **Meio de suporte:** Google Chrome Inspect para *debug*
- **Dispositivo de teste:** Xiaomi Redmi Note 4 Android version 6.0 MRA58K

O trabalho foi realizado no laboratório de informática da FEUP. O servidor utilizado foi clonado do repositório do WindsP no *GitHub* e foi colocado em execução dentro de uma máquina virtual, configurada com Vagrant, para o desenvolvimento e testes de execução.

5.2 Instalação

Antes de iniciar o desenvolvimento, é necessário instalar o Ionic e as suas dependências, nomeadamente o Cordova. Tanto o Ionic como o Cordova são instalados usando o pacote do Node.js, o **npm**.

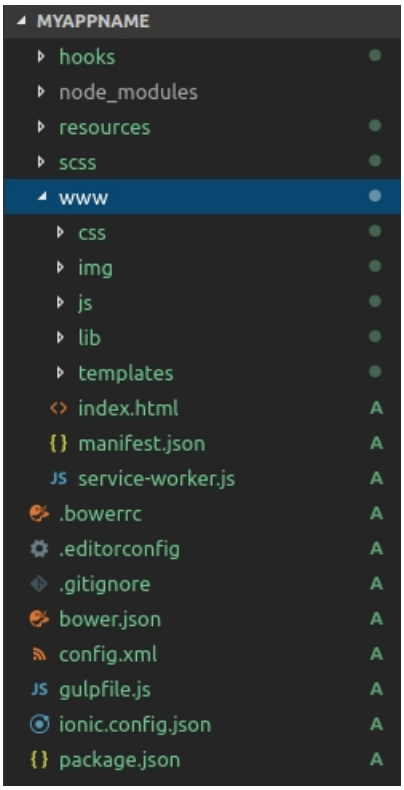

Figura 5.1: Organização padrão do código em Ionic

Concluída a instalação, inicia-se o projeto usando o *command line interface* (CLI) do Ionic. Ionic dispõe de três padrões para iniciar um projeto:

- ***blank*** — projeto sem nenhum *template*
- ***sidemenus*** — projeto com um menu lateral e que permite a navegação entre os *views* através desse menu.
- ***tabs*** — projeto com *tabs* na parte inferior do ecrã que permite acesso a múltiplos *views*

Escolheu-se iniciar o projeto com o *tabs*, executando o comando: ***ionic start myAppName tabs –type ionic1***.

5.3 Organização de Código

Quando se inicia um projeto, é gerada uma pasta com o nome atribuído aquando da criação do projeto (Figura 5.1).

Por omissão, o Ionic organiza os ficheiros da aplicação na pasta *WWW* da seguinte forma:

- ***css*** — contém os ficheiros CSS ou SCSS da aplicação.
- ***img*** — contém imagens que a aplicação usa.

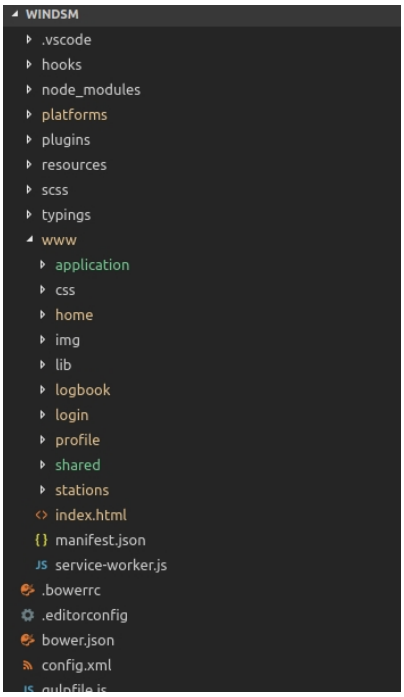

Figura 5.2: Organização do código utilizada no projeto

- ***js*** — contém os ficheiros JavaScript como *app.js, controllers.js e services.js*.
- ***lib*** — contém bibliotecas do Ionic e do AngularJS.
- ***templates*** — contém todos os *views*.
- ***index.html*** — página inicial que carrega todos os ficheiros JavaScript e CSS da aplicação.

Porém, à medida que a aplicação ganha maior dimensão, esta organização pode-se tornar pouco apropriada. Então, para uma melhor organização, foi decidido organizar o código por funcionalidades (Figura 5.2). Por exemplo na pasta ***logbook***, encontram-se todos os ficheiros desta funcionalidade: ***template***, ***controller***, ***modals***, ***services***.

5.4 Aspetos da Implementação

A aplicação é destinada apenas a utilizadores registados na plataforma WindsP. Por forma a permitir o acesso às suas funcionalidades é necessário autenticar-se com credenciais *email* e *password*. Essas credenciais são passadas à API do servidor que retorna um *token* em caso de sucesso. O *token* é armazenado e adicionado ao cabeçalho *Authorization* de pedidos HTTP (Listagem 5.1). O utilizador é então direcionado para o *view Home*. Em caso de erro na autenticação, é mostrada uma mensagem erro (Listagem 5.2).

```
1 .service('AuthService', function($q, $http,$rootScope, config, userAPI ) {
2     var LOCAL_TOKEN_KEY = 'yourTokenKey';
```

```
3     var isAuthenticated = false;
4     var authToken;
5     var currentUser = {};
6     function login(user, callback) {
7         return $http.post(config.baseUrl + 'auth/local', {
8             email: user.email,
9             password: user.password
10        }).then(function(res) {
11            storeUserCredentials(res.data.token);
12                }, function(err) {
13                logout();
14                return $q.reject(err.data);
15                });
16        };
17    function storeUserCredentials(token) {
18        window.localStorage.setItem(LOCAL_TOKEN_KEY, token);
19        useCredentials(token);
20     }
21    function useCredentials(token) {
22        isAuthenticated = true;
23        authToken = token;
24        $rootScope.$broadcast('AUTH_EXP');
25        $http.defaults.headers.common.Authorization = 'Bearer ' + authToken;
26    }
```

Listagem 5.1: *Login Service*

```
1 .controller('loginCtrl', function($scope, $state, $ionicPopup, AuthService) {
2     $scope.user = {
3         email: '',
4         password: ''
5     };
6     $scope.login = function() {
7
8         AuthService.login({
9             email: $scope.user.email,
10            password: $scope.user.password
11
12        }).then(function(msg) {
13            $state.go('tab.home');
14        }, function(errMsg) {
15            var msg;
16            if(errMsg == null)
17                msg = 'Sorry, the server is down!'
18            else
19                msg=errMsg.message;
20
21            var alertPopup = $ionicPopup.alert({
22                title: 'Login failed!',
23                template: msg
```

```
24              });
25          });
26      };
27  })
```

Listagem 5.2: *Login Controller*

A aplicação dispõe de quatro *tabs* que dão acesso aos *views* (Figura 5.3):

- ***Home*** — para visualizar a informação da experiência de campo.
- ***Logbook*** — para visualizar, criar, editar ou remover eventos.
- ***Stations*** — para visualizar o mapa com as estações instaladas no local da experiência.
- ***Profile*** — para visualizar o perfil do utilizador autenticado.

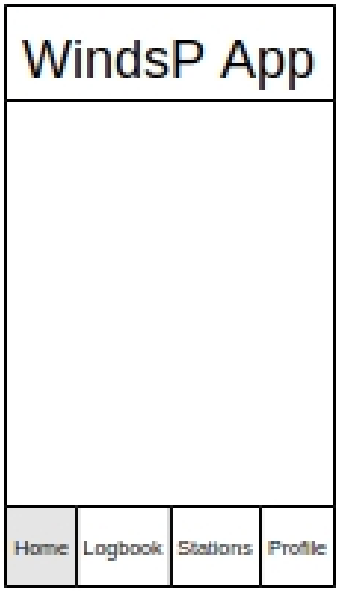

Figura 5.3: Modelo da aplicação

As rotas da aplicação estão definidas no ficheiro ***app.js*** na pasta ***application***. De notar que o *state tab* é a raiz de todos os outros *tabs*, sendo necessário declarar ***abstract: true*** (Listagem 5.3).

```
1  .config(function($stateProvider, $urlRouterProvider, $ionicConfigProvider) {
2     $stateProvider
3     .state('login', {
4               url: '/login',
5               templateUrl: 'login/login.html',
6               controller: 'loginCtrl'
7      })
8     .state('tab', {
9           url: '/tab',
```

```
10          abstract: true,
11          templateUrl: 'application/tabs.html'
12      })
13    .state('tab.home', {
14          url: '/home',
15          views: {
16              'tab-home': {
17                  templateUrl: 'home/home.html',
18                  controller: 'homeCtrl' }
19          }
20      })
21     .state('tab.logbook', {
22          url: '/logbook',
23          views: {
24              'tab-logbook': {
25                  templateUrl: 'logbook/logbook.html',
26                  controller: 'logbookCtrl' }
27          }
28      })
29      .state('tab.station', {
30          url: '/station',
31          views: {
32              'tab-station': {
33                  templateUrl: 'stations/station.html',
34                  controller: 'stationCtrl' }
35          }
36      })
37      .state('tab.profile', {
38          url: '/profile',
39          views: {
40              'tab-profile': {
41                  templateUrl: 'profile/profile.html',
42                  controller: 'profileCtrl' }
43          }
44      })
45    $urlRouterProvider.otherwise('/login');
46  })
```

Listagem 5.3: Rotas da aplicação

As solicitações ao servidor são feitas utilizando o serviço ***$http*** do AngularJS. Foram criados serviços do tipo *factory* onde são feitas solicitações ao servidor (Listagem 5.4). Esses serviços são injetados nos controladores que têm acesso a todos os métodos do serviço.

```
1       .factory('logbookAPI', function($http, config) {
2           var _getEvent = function() {
3               return $http.get(config.baseUrl + 'api/events?experiment_id=3').then(
                    function(res){
4               return res;
5               });
```

```
6         };
7         var _getEventById = function(id) {
8             return $http.get(config.baseUrl + 'api/events/' + id).then(function(res
                  ){
9             return res;
10            });
11        };
12        var _postEvent = function(event) {
13            return $http.post(config.baseUrl + 'api/events', event).then(function(
                  res){
14            return res;
15            });
16        };
17        var _delEvent = function(id) {
18            return $http.delete(config.baseUrl + 'api/events/' + id).then(function(
                  res){
19            return res;
20            });
21        };
22        var _putEvent = function(id, event) {
23            return $http.put(config.baseUrl + 'api/events/' + id, event).then(
                  function(res){
24            return res;
25            });
26        };
27        return {
28            getEvent: _getEvent,
29            postEvent: _postEvent,
30            delEvent: _delEvent,
31            putEvent: _putEvent,
32            getEventById: _getEventById
33        };
34    });
```

Listagem 5.4: *Logbook API Service*

No *tab Logbook* podem-se filtrar os eventos por categorias (*alert*, *issue*, *meeting*, etc.), palavras e data. A filtragem por palavras é feita utilizando o *filter* padrão do AngularJS. A filtragem por data requer a utilização dum filtro personalizado. O filtro recebe como parâmetro a data de início e fim e compara-as com a data em que o evento foi criado. Se as datas da criação dos eventos estiverem nesse intervalo, o filtro retorna um objeto com esses eventos (Listagem 5.5).

```
1  .filter('dateRange', function() {
2         return function(items, startDate, endDate) {
3             var filtered = [];
4             var date;
5             if (!startDate && !endDate || !endDate) {
6                 return items;
7             }
```

```
8             angular.forEach(items, function(item) {
9                 date = new Date(item.created_at);
10                date.setHours(0, 0, 0, 0);
11                if (date >= startDate && date <= endDate) {
12                   filtered.push(item);
13                }
14            });
15           return filtered;
16        }
17    })
```

Listagem 5.5: Filtro personalizado de data

Para a aplicação aceder ao calendário do dispositivo e escolher uma data, foi utilizado o *plugin* do Cordova ***$cordovaDatePicker***.

Ao criar um evento, pode adicionar-se a localização em que o utilizador se encontra, assim como fotos. O utilizador pode optar por tirar uma nova foto ou então selecionar fotos da galeria. Para o acesso à geo-localização e à câmara/galeria, foram utilizados os *plugins* ***$cordovaGeolocation*** e ***$cordovaCamera***, respetivamente. Para o *upload* das fotos no servidor utilizou-se o *plugin* ***$cordovaFileTransfer***.

No *tab Stations* é possível visualizar o mapa com as estações. As estações são representadas por marcadores personalizados com *ícones* das respetivas categorias. Foi utilizado o *Google Maps JavaScript API* para representar o mapa. Após a criação do mapa (*function initMap()*), é chamada a *function createMarkers*, que é responsável por gerar os marcadores (*markers*) (Listagem 5.6).

```
1  function initMap() {
2          var latPLngP = { lat: 39.701301, lng: -7.735229 };
3          var mapOptions = {
4              center: latPLngP,
5              zoom: 13,
6              mapTypeId: google.maps.MapTypeId.SATELLITE
7          };
8          map = new google.maps.Map(document.getElementById("map"), mapOptions);
9          google.maps.event.addListenerOnce(map, 'idle', function() {
10             createMarkers();
11         });
12     }
13 function createMarkers() {
14          angular.forEach(stations, function(stations) {
15             var coordinates = getCoordinatesIn(stations, 'WGS84');
16             var latLng = new google.maps.LatLng(coordinates.y, coordinates.x);
17             var icon = config.baseUrl + 'api/documents/noVersionFiles/' + stations.
                   equipmentSubcategory.icon;
18             var labels = stations.number;
19             var equipmentSubcategory = stations.equipment_subcategory_id;
20             var marker = new google.maps.Marker({
21                 map: map,
22                 icon: icon,
```

```
23                label: {
24                    text: "" + labels,
25                    color: 'white' },
26                position: latLng,
27                equipmentSubcategory: equipmentSubcategory
28            });
29            markers.push(marker);
30            google.maps.event.addListener(marker, 'click', function() {
31                   stationById(stations.id, openStationModal);
32            });
33        });
34    }
```

Listagem 5.6: Mapa e marcadores

As estações podem ser filtradas pelas suas categorias ou pela localização dentro do raio introduzido pelo utilizador. Neste último caso, é obtida a posição em que o utilizador se encontra e calcula-se a distância das estações relativamente à posição. Se a distância for igual ou menor ao raio introduzido, as estações são mostradas. Foi utilizada a função ***computeDistanceBetween*** da biblioteca de geometria da *Google Maps*, que permite calcular a distância entre dois pontos recebendo como parâmetro as coordenadas geodésicas (Listagem 5.7).

```
1  $scope.getMarkersInRadius = function(radius) {
2          angular.forEach(markers, function(markers) {
3              var distance = google.maps.geometry.spherical.computeDistanceBetween(
                   markers.position, myPosition);
4              if (distance <= radius) {
5                  markers.setVisible(true);
6              } else {
7                  markers.setVisible(false);
8              }
9          });
10     }
```

Listagem 5.7: Estações num raio

Para além de obter a localização do utilizador, o *plugin* ***$cordova Geolocation*** foi utilizado para o seguir à medida que se desloca.

5.5 Teste

Foram feitos apenas testes tradicionais de *debug*. O Ionic permite testes diretamente no *browser*. Tendo em conta que a aplicação tem acesso à câmara, sensor GPS e calendário, não era possível testá-los no *browser*, sendo para isso necessário efetuar testes num emulador ou num dispositivo móvel.

Os testes foram efetuados num dispositivo Android. Através do comando ***ionic cordova run android*** e com o dispositivo no modo *developer* instalou-se a aplicação. Para a visualização de eventuais erros utilizou-se o *Google inspect* (figuras 5.4 e 5.5).

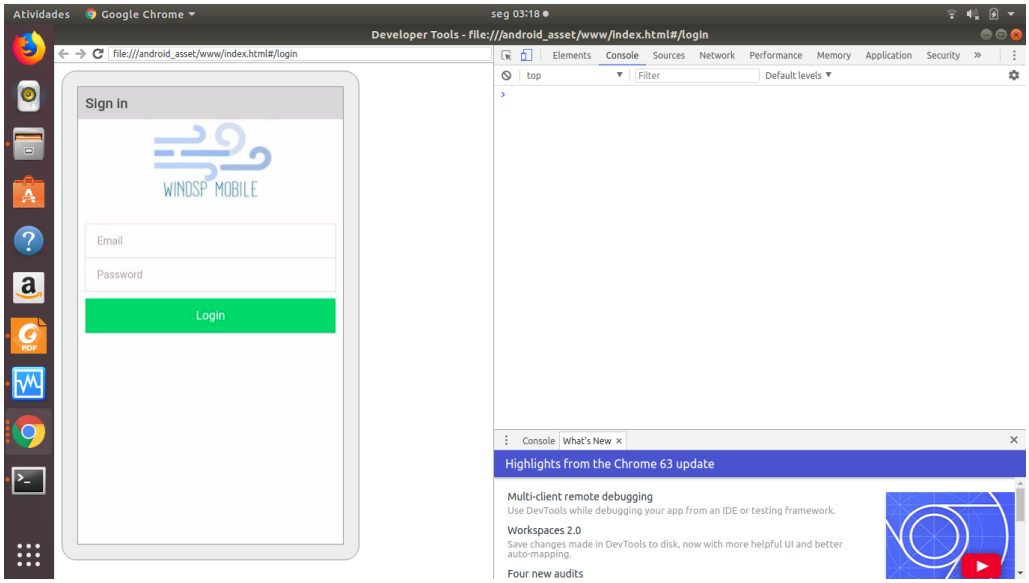

Figura 5.4: *Google inspect 1*

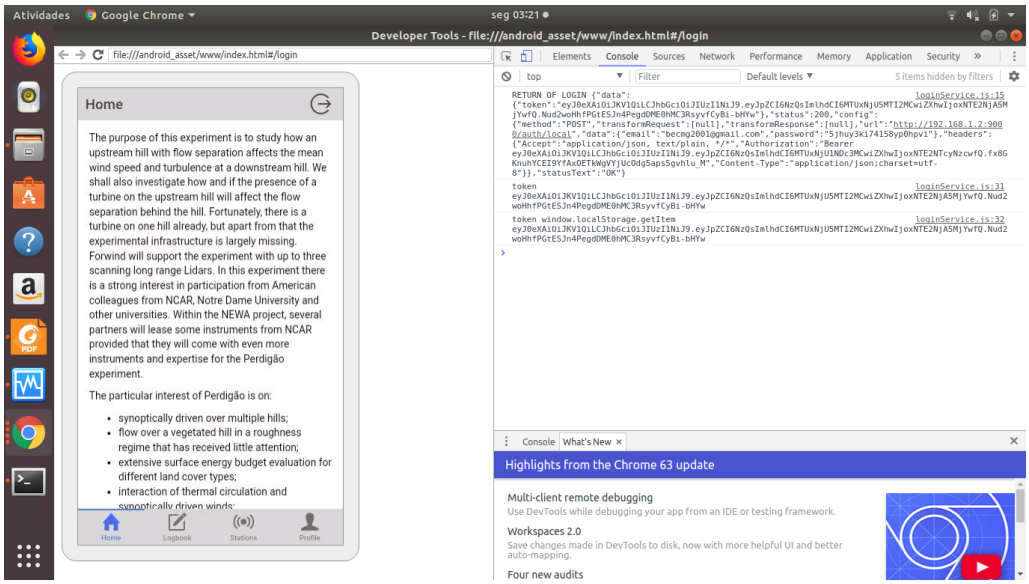

Figura 5.5: *Google inspect 2*

5.6 Resumo

Este capítulo caracteriza o ambiente de desenvolvimento e apresenta os detalhes técnicos relativos à conceção da aplicação. São ainda abordados aspetos relativos à instalação da aplicação e organização do código. No Capítulo 6 vão ser apresentados os resultados da aplicação.

Capítulo 6

Resultados

Neste capítulo apresentam-se os resultados em forma de casos de utilização, ilustrados com figuras — *screenshots* da aplicação a correr num *smarthphone* com sistema Android. Para o sistema iOS só são apresentadas figuras de casos cuja *interface* difere da do Android. As figuras do sistema iOS foram extraídas do emulador do IDE Xcode.

6.1 Autenticação

Ao iniciar a aplicação é necessário autenticar-se (Figura 6.1). No caso de introdução de credenciais erradas, o sistema alerta através de uma mensagem de erro (figuras 6.2, 6.3).

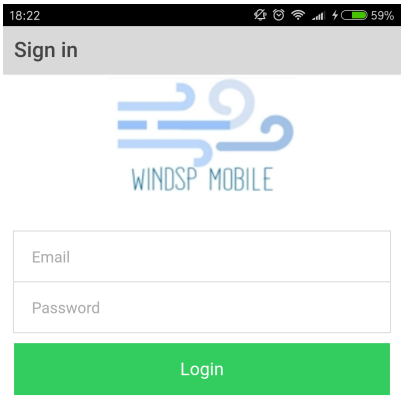

Figura 6.1: *Login*

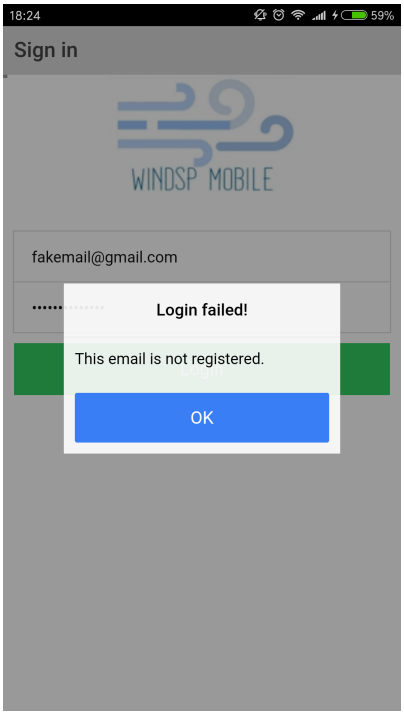

Figura 6.2: *Login error 1*

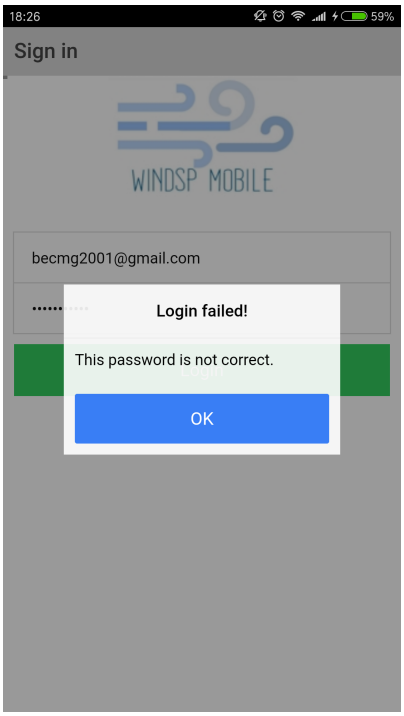

Figura 6.3: *Login error 2*

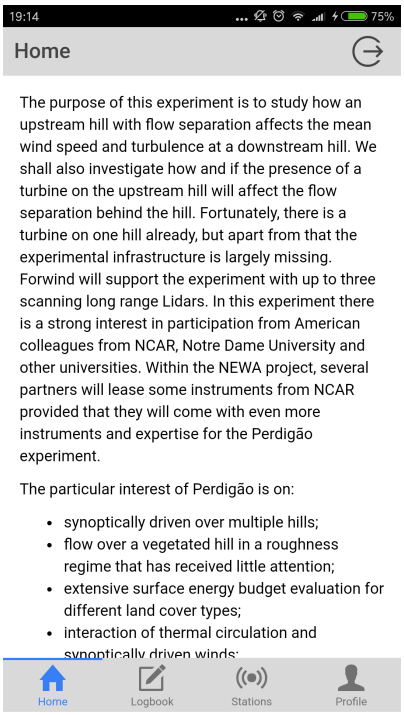

Figura 6.4: *Home view*

6.2 Home

Com o *login* bem sucedido, o utilizador é encaminhado para o *view Home* onde se encontram informações sobre o objetivo da experiência (Figura 6.4).

6.3 Logbook e Gestão de Eventos

Através da *tab Logbook* ganha-se acesso a *view Logbook* onde se pode visualizar, criar, editar e remover eventos (figuras 6.5, 6.6, 6.7, 6.15, 6.16, 6.17, 6.19).

Tendo em conta a existência de vários eventos, o utilizador, querendo encontrar um evento específico, pode fazer filtragem por palavras chave, categoria e data do evento (figuras 6.8, 6.9, 6.10, 6.13, 6.11, 6.12, 6.14).

Para criar um novo evento, o utilizador deve clicar no ícone **+** no canto superior direito. É aberto um *modal* para preenchimento do formulário (Figura 6.15). Neste formulário o utilizador pode adicionar fotos e a sua geo-localização quando criar o evento (Figura 6.16). Caso tenha criado algum evento (Figura 6.17), o utilizador pode editá-lo ou então removê-lo (figuras 6.18, 6.19). Também existe a possibilidade de visualizar o perfil dos autores dos eventos.

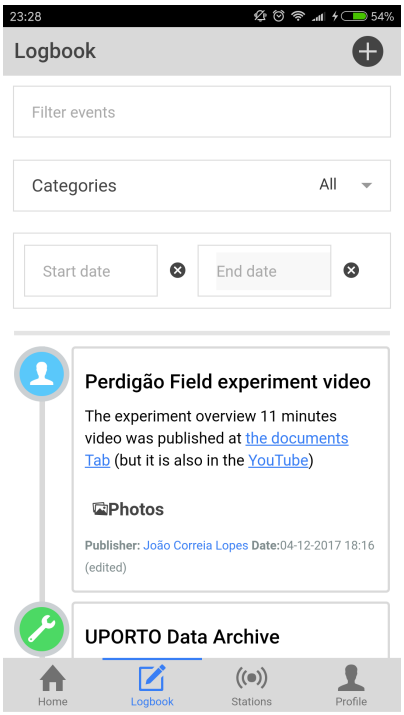

Figura 6.5: *Logbook view 1*

Figura 6.6: *Logbook view 2*

6.4 Estações — Visualização e filtragem

A *tab Stations* mostra o mapa com as estações da experiência (Figura 6.20). As estações podem ser filtradas por categoria de estações ou pela distância a que o utilizador se encontra delas (figuras 6.21, 6.22, 6.23, 6.24, 6.25, 6.26).

Ao carregar num ícone de uma determinada estação, abre-se um *modal* com a informação desta estação (figuras 6.27, 6.28). Carregando numa foto da estação, abre-se um outro *modal* que permite visualizar a imagem ampliada (Figura 6.29). Caso o utilizador queira adicionar uma foto à estação, deverá carregar no botão *Add Photo* (Figura 6.28). Também é possível seguir a posição do utilizador à medida que se desloca (Figura 6.30)

6.5 Perfil

Por último, a *tab Profile* mostra o perfil do utilizador autenticado na aplicação (Figura 6.31).

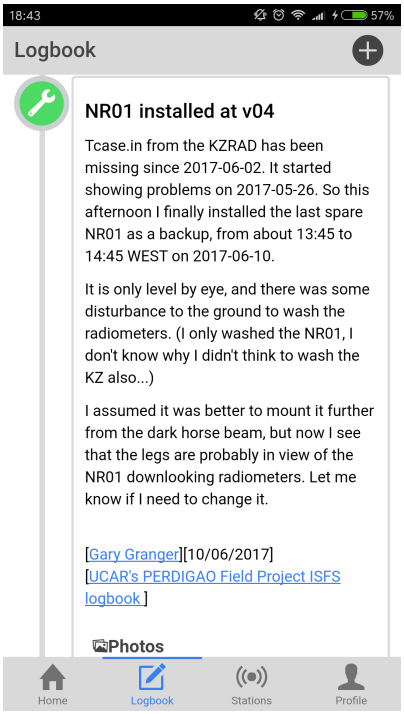

Figura 6.7: *Logbook view 3*

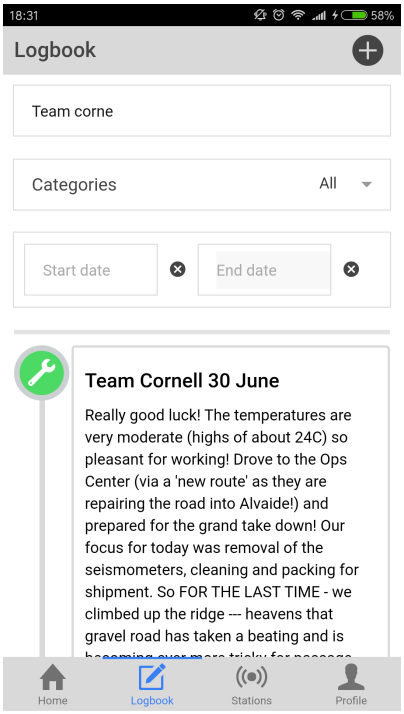

Figura 6.8: *Events filtered by words*

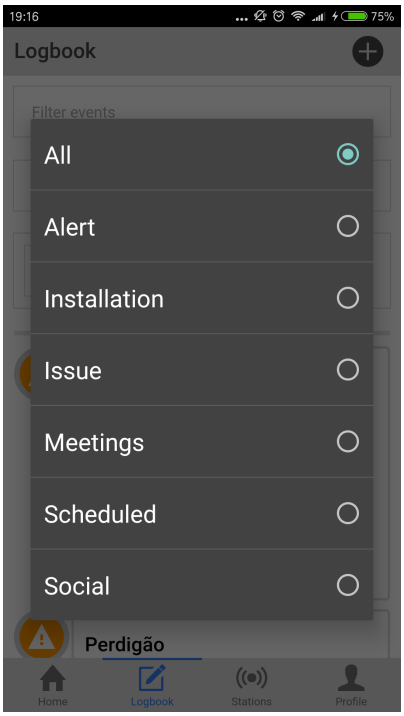

Figura 6.9: *Logbook categories*

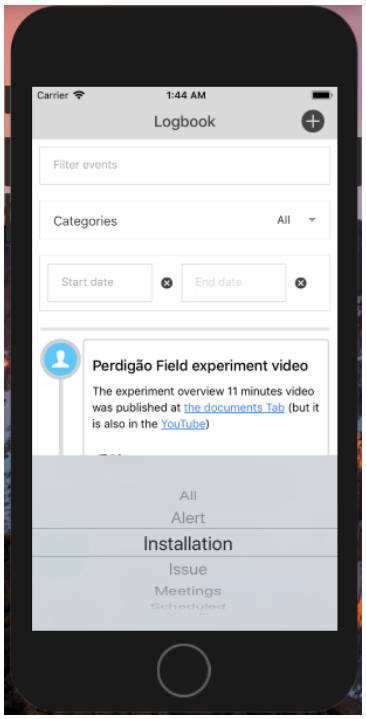

Figura 6.10: *Logbook categories (iOS)*

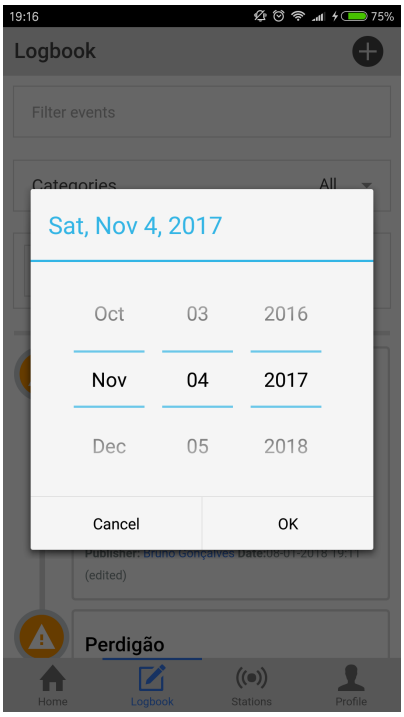

Figura 6.11: *Logbook date picker*

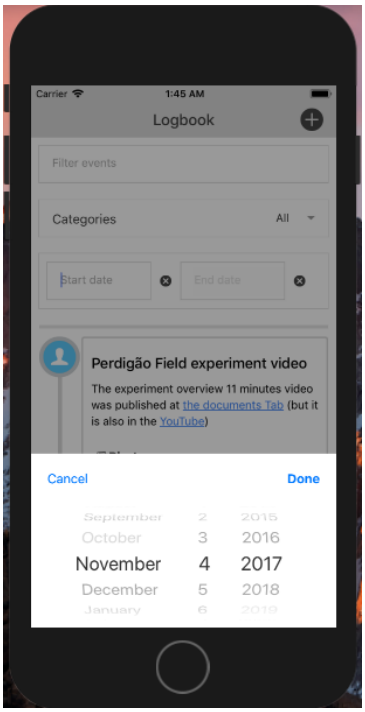

Figura 6.12: *Logbook date picker (iOS)*

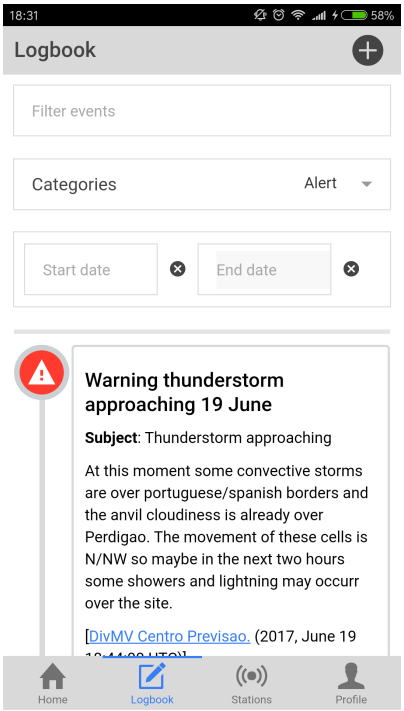

Figura 6.13: *Alert events*

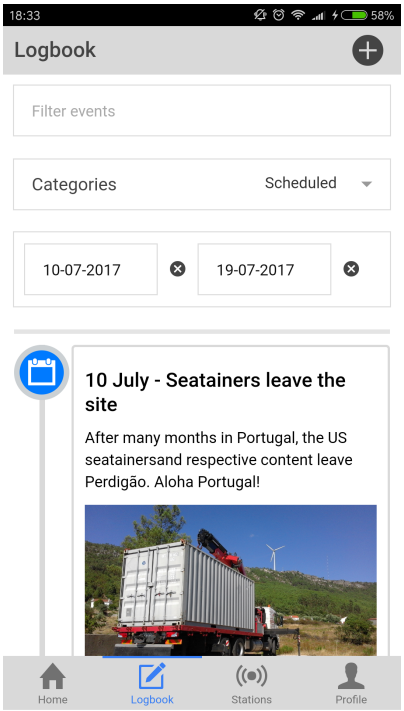

Figura 6.14: *Scheduled events*

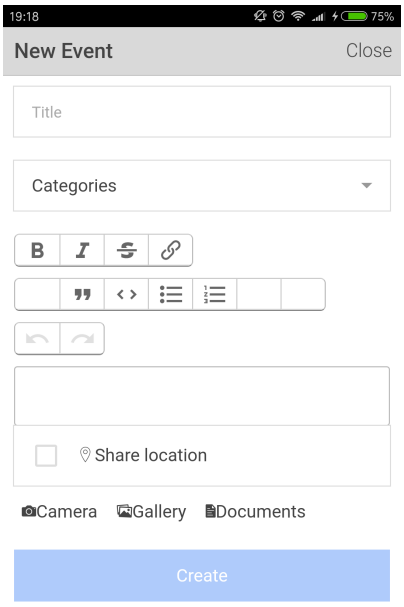

Figura 6.15: *New event*

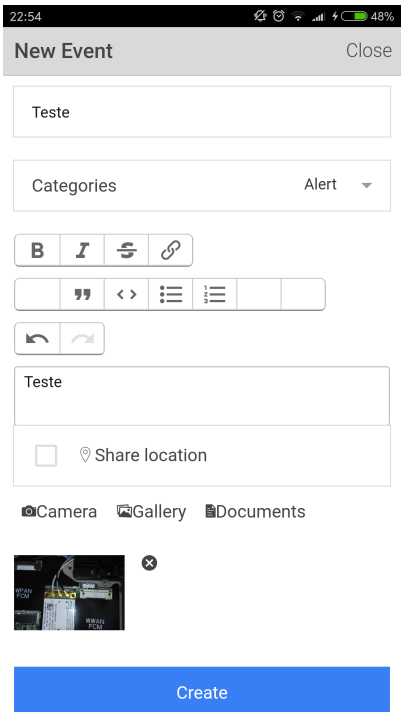

Figura 6.16: *Creating event*

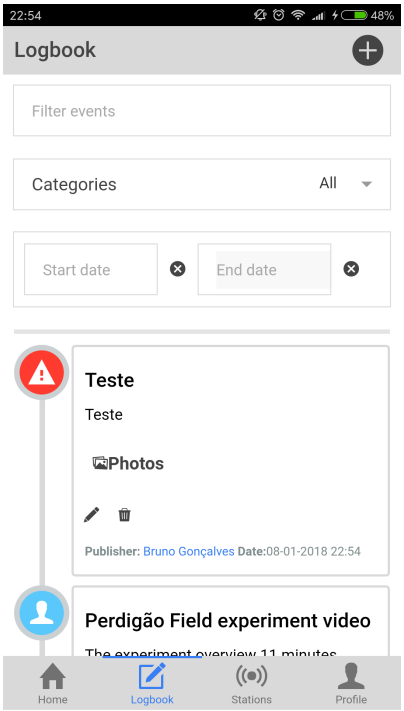

Figura 6.17: *Event created*

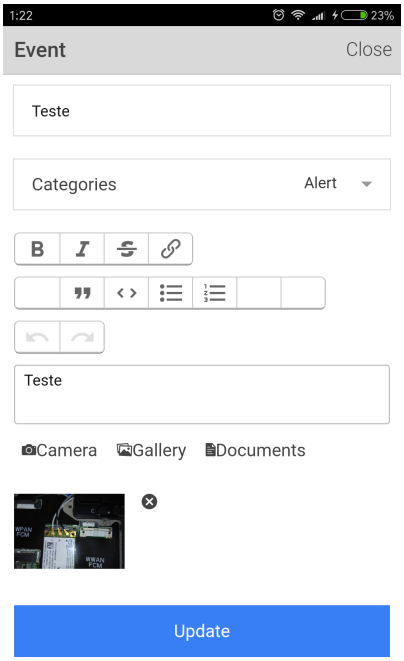

Figura 6.18: *Editing event*

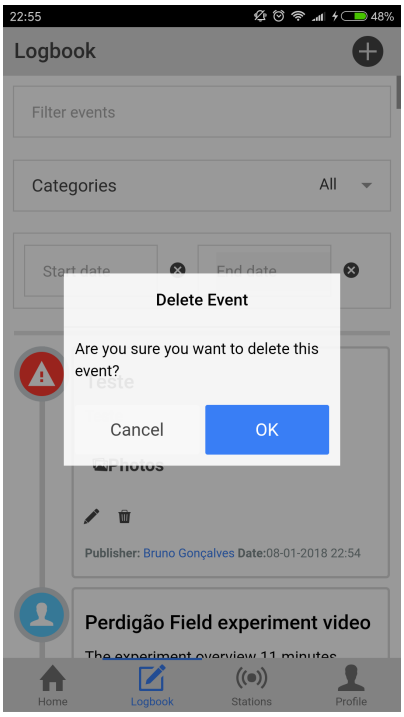

Figura 6.19: *Deleting event*

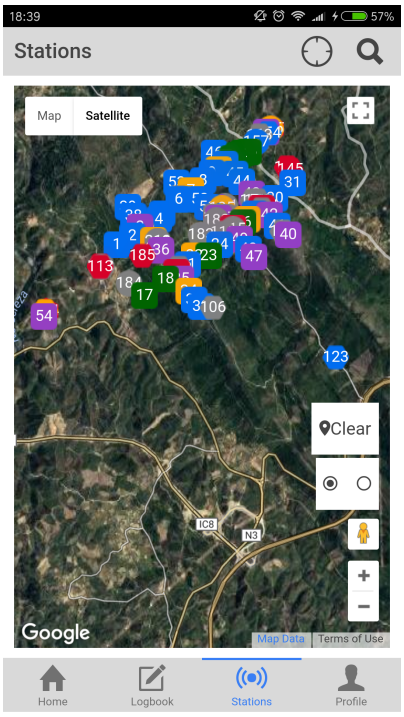

Figura 6.20: *Stations map*

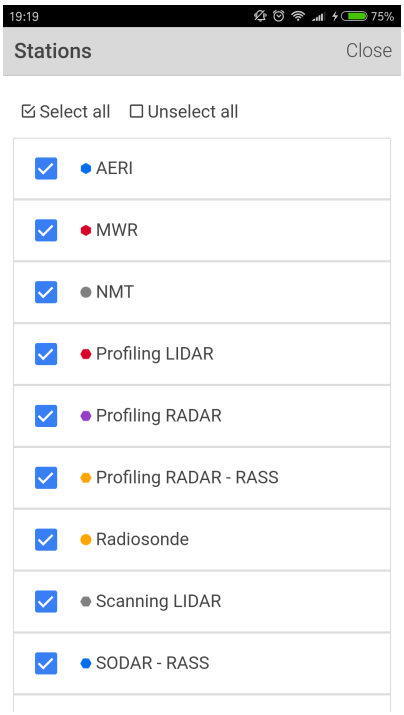

Figura 6.21: *Stations categories filter*

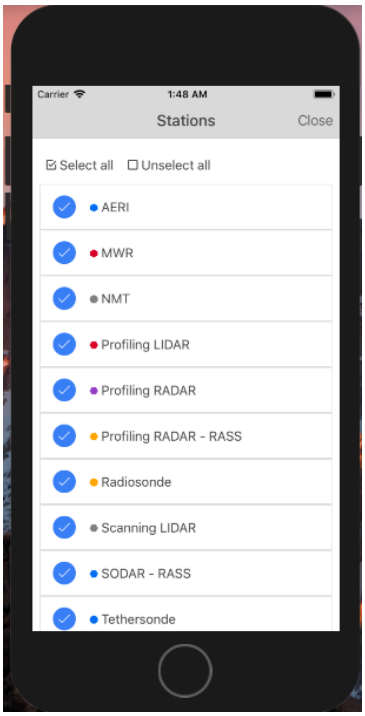

Figura 6.22: *Stations categories filter (iOS)*

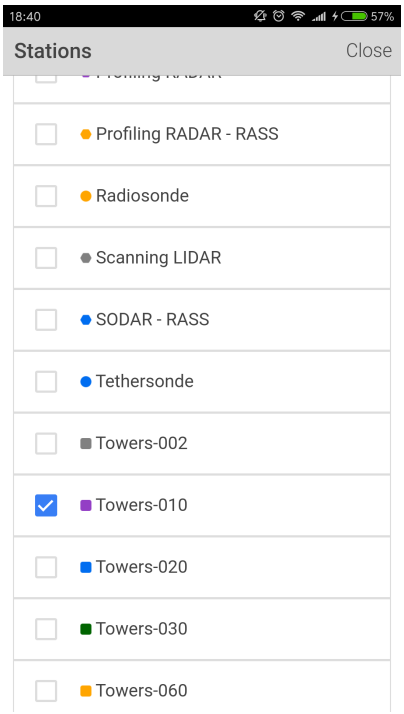

Figura 6.23: *Selecting towers-010 stations*

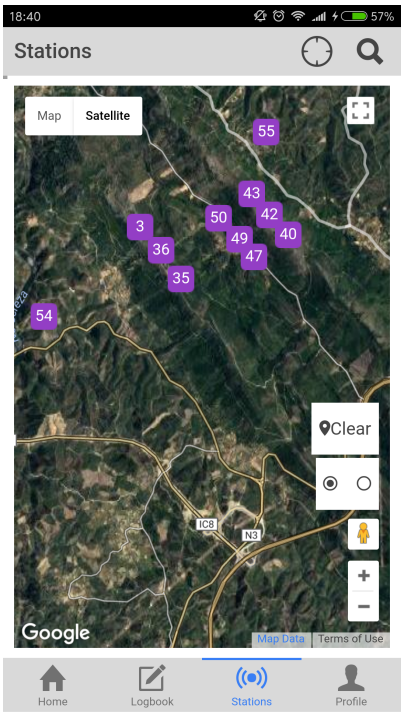

Figura 6.24: *Towers-10 stations*

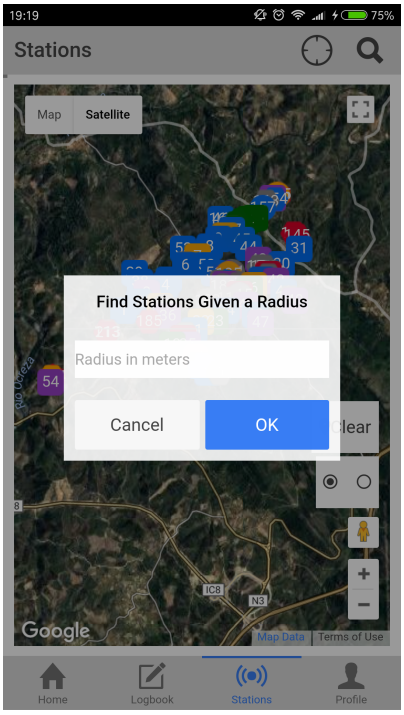

Figura 6.25: *Filtering by radius*

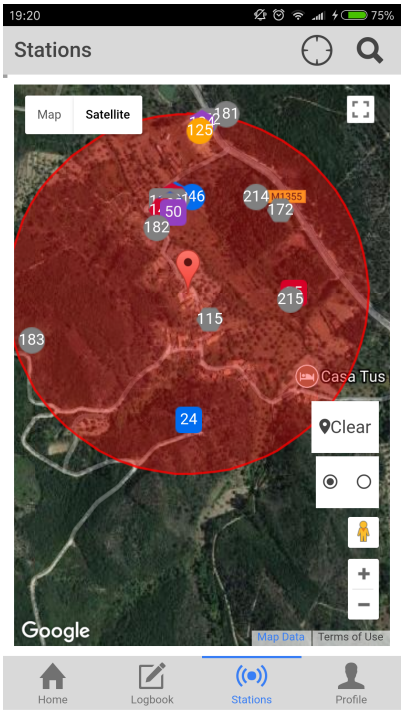

Figura 6.26: *Stations in radius*

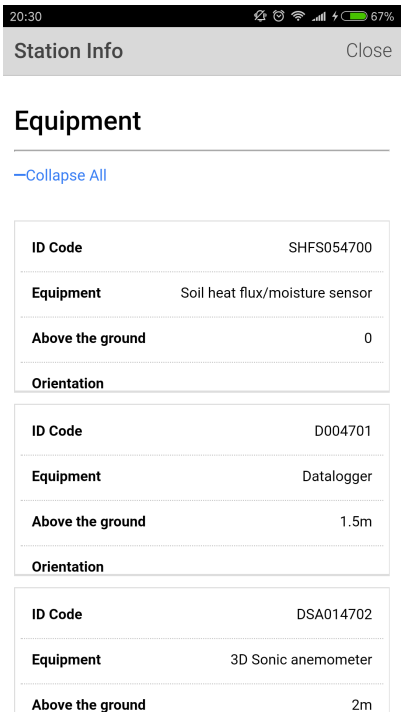

Figura 6.27: *Station info 1*

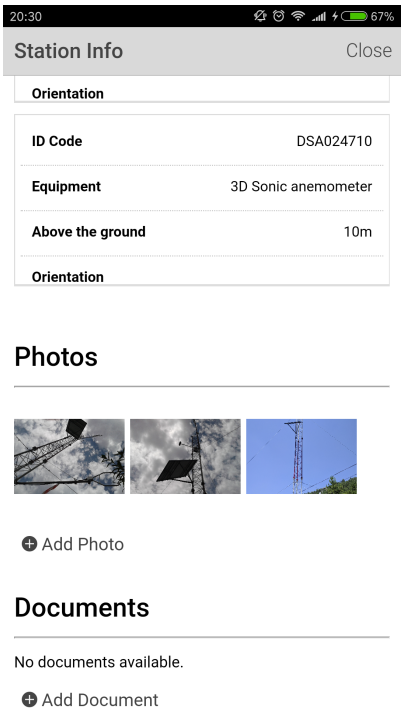

Figura 6.28: *Station info 2*

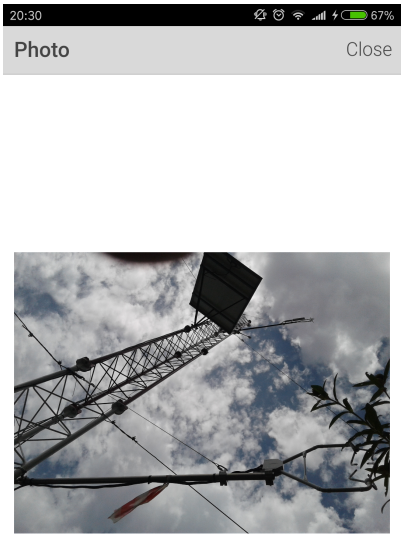

Figura 6.29: *Photo modal view*

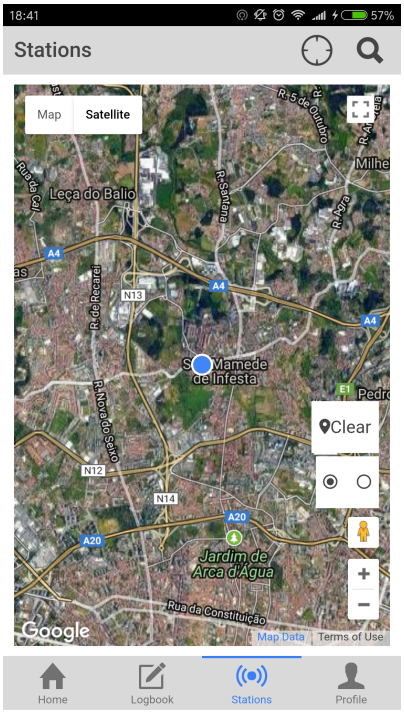

Figura 6.30: *Tracking position*

Figura 6.31: *Profile view*

Capítulo 7

Conclusões

Este trabalho surge no âmbito do projeto NEWA que visa estudar as características do vento na Europa de modo a produzir uma base de dados, quantificar o potencial energético eólico, construir modelos para o desenvolvimento de mapas de vento e, desta forma, aumentar o aproveitamento eficiente da energia limpa.

Para atingir esses objetivos são realizadas experiências (campanhas de campo). Uma campanha decorreu em Perdigão onde foi instalada uma quantidade significativa de estações de recolha de dados, tornando difícil para o investigador no terreno localizar as estações bem como os equipamentos instalados nas torres.

7.1 Trabalho Efetuado

O propósito do trabalho é desenhar, implementar e testar uma aplicação móvel para gestão de eventos da campanha a decorrer em Perdigão, apoiando os investigadores que se encontram no terreno e permitido-lhes tirar partido das funcionalidades do telemóvel, como o sensor GPS e câmara fotográfica. Com a aplicação móvel e o auxílio do GPS, os investigadores poderão pesquisar as estações que se encontram à sua volta e os respetivos dispositivos instalados. Eles poderão, ainda, registar eventos, tais como, falhas de torres e/ou equipamentos, condições físicas à volta das estações, e reportá-los imediatamente com eventual integração de fotografias.

Começou-se por fazer um estudo dos tipos de aplicações e dos *frameworks* de desenvolvimento. De seguida, foram definidos os requisitos da aplicação tendo como base a reunião tida com os *stakeholders*.

A abordagem híbrida foi considerada a melhor solução para o caso em apreço e o *framework* Ionic foi selecionado para a implementação. O Ionic é referenciado pelas suas enormes potencialidades, vantagens da sua programação ser baseada em linguagem JavaScript, flexibilidade, estar bem documentada e ter uma grande comunidade *online*.

Com a implementação de um protótipo cumpriram-se os requisitos fixados, ou seja, a aplicação móvel permite gerir eventos, pesquisar as estações e tirar partido dos recursos do telemóvel sensor

GPS e câmara. O funcionamento da aplicação no sistema operativo Android e iOS foi confirmado através de testes num dispositivo móvel, caso do Android, e num emulador para o iOS.

Com a criação desta aplicação para *smartphone* tornaram-se evidentes as muitas vantagens associadas à portabilidade do equipamento. Os investigadores, doravante, podem, a qualquer momento e sítio, aceder às informações relevantes. Os investigadores poderão utilizar os seus próprios aparelhos móveis uma vez que a aplicação funciona independentemente da plataforma móvel.

7.2 Trabalho Futuro

A solução implementada foi apenas um caso de estudo que servirá de suporte para implementações futuras das funcionalidades seguintes:

- **Funcionamento *offline*** — Tendo em conta o local da experiência, onde pode faltar rede, será interessante a pesquisa da informação das estações funcionar no modo *offline*. Para tal, será necessário ter uma base de dados no telemóvel onde se armazena a informação das estações.
- **Notificações** — Receber notificações aquando da criação de um evento. O utilizador poderá personalizar as notificações pelas categorias, por exemplo, receber notificações apenas de eventos da categoria *danger*.
- **Código QR** — Integração de um *scanner* de códigos QR com a aplicação. Cada torre terá um código QR que, quando verificado, o utilizador terá acesso à informação da torre, por exemplo, aos equipamentos nela instalada.

Os testes feitos à aplicação foram testes tradicionais de *debug*, sendo que no futuro será necessário levar a cabo testes unitários para facilitar a evolução da app (introdução de melhorias e correção de *bugs*).

Referências

[1] NEWA. NEWA Call for Wind Data. 2015.

[2] Tony Hey, Stewart Tansley, Kristin M Tolle, et al. *The fourth Paradigm: Data-Intensive Scientific Discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[3] Daniel Atkins, Christine Borgman, Nathan Bindoff, Mark Ellisman, Stuart Feldman, Ian Foster, Albert Heck, Dieter Heermann, Julia Lane, Luciano Milanesi, et al. Building a UK Foundation for the Transformative Enhancement of Research and Innovation. *Research Councils UK*, 2009.

[4] José Tiago Paiva Antunes Magalhães. Suporte para Séries Temporais em Plataforma e-Science. Tese de mestrado, Faculdade de Engenharia da Univesidade do Porto, 2015.

[5] Windsp. Manual de especificação. http://winds.fe.up.pt/wiki/doku.php/windsp/docv02/index. Acedido: 2017-09-18.

[6] N. Veretennikova, V. Pasichnyk, N. Kunanets, e B. Gats. E-science: New paradigms, system integration and scientific research organization. Em *2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies"(CSIT)*, páginas 76–81, Sept 2015.

[7] Shannon Bohle. What is E-science and How Should it be Managed? *Nature.com, Spektrum der Wissenschaft (Scientific American)*, (June) 2013.

[8] Windsp. Manual de utilizador. http://winds.fe.up.pt/wiki/doku.php/windsp/usermanual-v023/index. Acedido: 2017-09-18.

[9] Alexandra Witze. World's largest wind-mapping project spins up in portugal. *Nature*, Dec 2017. URL: https://www.nature.com/news/world-s-largest-wind-mapping-project-spins-up-in-portugal-1.21481.

[10] Harindra J. S. Fernando, Julie K. Lundquist, e Steven Oncley. Monitoring wind in portugal's mountains down to microscales. *Eos*, May 2017. URL: https://eos.org/project-updates/monitoring-wind-in-Portugals-mountains-down-to-microscales.

[11] Spyros Xanthopoulos e Stelios Xinogalos. A comparative analysis of cross-platform development approaches for mobile applications. Em *Proceedings of the 6th Balkan Conference in Informatics*, páginas 213–220. ACM, 2013.

[12] Wafaa S. El-Kassas, Bassem A. Abdullah, Ahmed H. Yousef, e Ayman M. Wahba. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, 8(2):163 – 190, 2017. URL: http://www.sciencedirect.com/science/article/pii/S2090447915001276, doi:https://doi.org/10.1016/j.asej.2015.08.004.

[13] T. Freitas Bernardes e M. Yoshikazu Miyake. Cross-platform mobile development approaches: A systematic review. *IEEE Latin America Transactions*, 14(4):1892–1898, April 2016.

[14] Mounaim Latif, Younes Lakhrissi, El Habib Nfaoui, e Najia Es-Sbai. Cross platform approach for mobile application development: A survey. Em *Information Technology for Organizations Development (IT4OD), 2016 International Conference on*, páginas 1–5. IEEE, 2016.

[15] Henning Heitkötter, Sebastian Hanschke, e Tim A. Majchrzak. Evaluating cross-platform development approaches for mobile applications. Em José Cordeiro e Karl-Heinz Krempels, editores, *WEBIST (Selected Papers)*, volume 140 de *Lecture Notes in Business Information Processing*, páginas 120–138. Springer, 2012.

[16] IBM. Native, web or hybrid mobile-app development, 2012. URL: ftp://public.dhe.ibm.com/software/pdf/mobile-enterprise/WSW14182USEN.pdf.

[17] Pavel Smutnỳ. Mobile development tools and cross-platform solutions. Em *Carpathian Control Conference (ICCC), 2012 13th International*, páginas 653–656. IEEE, 2012.

[18] WorkLight Webinar Series. Native, web or hybrid mobileapp development? http://www.slideshare.net/SuraGonzalez/wl-native-vshybrid. Acedido: 2017-02-03.

[19] Lionbridge. Mobile Web Apps vs. Mobile Native Apps: How to Make the Right Choice. 2012.

[20] Mohamed LACHGAR e Abdelmounaïm ABDALI. Decision Framework for Mobile Development Methods. *International Journal of Advanced Computer Science and Applications(ijacsa)*, 2017.

[21] Xamarin. https://www.xamarin.com/platform. Acedido: 2017-02-03.

[22] Manuel Palmieri, Inderjeet Singh, e Antonio Cicchetti. Comparison of cross-platform mobile development tools. Em *Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on*, páginas 179–186. IEEE, 2012.

[23] Paulo RM de Andrade, Adriano B Albuquerque, Otávio F Frota, Robson V Silveira, e Fátima A da Silva. Cross platform app: a comparative study. *arXiv preprint arXiv:1503.03511*, 2015.

[24] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edição, 2010.

[25] Rodrigues da Silva Alberto Manuel e Escaleira Videira Carlos Alberto. *UML, metodologia e ferramentas CASE*. Centro Atlantico, 2001.

[26] Scott W. Ambler. *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press, New York, NY, USA, 3rd edição, 2004.

[27] Mike Cohn. User stories and user story examples by Mike Cohn. URL: https://www.mountaingoatsoftware.com/agile/user-stories.

[28] Karl Eugene Wiegers. *Software Requirements*. Microsoft Press, Redmond, WA, USA, 2 edição, 2003.

[29] AngularJS. Develope Guide. https://docs.angularjs.org/guide/introduction. Acedido: 2017-11-10.

[30] AngularJS. Data binding. https://docs.angularjs.org/guide/databinding. Acedido: 2017-11-10.

[31] R. Branas. *AngularJS Essentials*. Community Experience Distilled. Packt Publishing, 2014.

[32] A. Osmani. *Learning JavaScript Design Patterns*. O'Reilly Media, 2012. URL: https://www.safaribooksonline.com/library/view/learning-javascript-design/9781449334840/.

[33] Apache. Apache cordova. https://cordova.apache.org/docs/en/latest/guide/overview/index.html. Acedido: 2017-02-03.

[34] Ionic. Structure. https://ionicframework.com/docs/v1/concepts/structure.html. Acedido: 2017-02-03.